GETTING STARTED

EDIT

Welcome to the WebdriverIO documentation. It will help you to get started fast. If you run into problems you can find help and answers on our Gitter Channel or you can hit me on Twitter.

Also, if you encounter problems in starting up the server or running the tests after following this tutorial, ensure that the server and the geckodriver are listed in your project directory. If not, re-download them per steps 2 and 3 below.

> **Note:** These are the docs for the latest version (>= v5.0.0) of WebdriverIO. If you are still using v4 or older please use the legacy docs website v4.webdriver.io!

The following will give you a short step by step introduction to get your first WebdriverIO script up and running.

Taking the first step

Let's suppose you have Node.js already installed. If you don't have Node installed, we recommend installing NVM to assist managing multiple active Node.js versions.

First thing we need to do is to download a browser driver that helps us automate the browser. To do so we create an example folder first:

Create a simple test folder

```
$ mkdir webdriverio-test && cd webdriverio-test
```

While still in this test folder:

Download Geckodriver

Note: You must have Firefox installed to use Geckodriver.

Download the latest version of geckodriver for your environment and unpack it in your

```
$ curl -L https://github.com/mozilla/geckodriver/releases/download/v0.21.0/geckodriver
```

OSX

```
$ curl -L https://github.com/mozilla/geckodriver/releases/download/v0.21.0/geckodriver
```

Windows 64 bit

Simple setup: (Chocolatey)

```
choco install selenium-gecko-driver
```

For advanced users (Powershell):

```
# Run as privileged session. Right-click and set 'Run as Administrator'
# Use geckodriver-v0.21.0-win32.zip for 32 bit Windows
$url = "https://github.com/mozilla/geckodriver/releases/download/v0.21.0/geckodriver-v
$output = "geckodriver.zip" # will drop into current directory unless defined otherwis
$unzipped_file = "geckodriver" # will unzip to this folder name

# By default, Powershell uses TLS 1.0 the site security requires TLS 1.2
[Net.ServicePointManager]::SecurityProtocol = [Net.SecurityProtocolType]::Tls12

# Downloads Geckodriver
Invoke-WebRequest -Uri $url -OutFile $output

# Unzip Geckodriver
Expand-Archive $output -DestinationPath $unzipped_file
cd $unzipped_file

# Globally Set Geckodriver to PATH
[System.Environment]::SetEnvironmentVariable("PATH", "$Env:Path;$pwd\geckodriver.exe",
```

Note: Other geckodriver releases are available here. In order to automate other browser you need to run different drivers. You can find a list with all drivers in the awesome-

```
$ /path/to/binary/geckodriver --port 4444
```

For example, if you ran the curl command from above, you should have a `geckodriver` binary available in the current folder. You can run the following to start it:

```
$ ./geckodriver --port 4444
```

This will start Geckodriver on `localhost:4444` with the WebDriver endpoint set to `/`.

Keep this running in the background and open a new terminal window. Next step is to download WebdriverIO via NPM:

Download WebdriverIO

By calling:

```
$ npm install webdriverio
```

Create Test File

Create a test file (e.g. `test.js`) with the following content:

```
const { remote } = require('webdriverio');

(async () => {
    const browser = await remote({
        logLevel: 'error',
        path: '/',
        capabilities: {
            browserName: 'firefox'
        }
    });

    await browser.url('https://webdriver.io');
```

```
const title = await browser.getTitle();
console.log('Title was: ' + title);
```

```
})().catch((e) => console.error(e));
```

Run your test file

Make sure you have at least Node.js v8.11.2 or higher installed. To update your current running Node.js version install nvm and follow their instructions. Once that is done run the test script by calling:

```
$ node test.js
```

this should output the following:

```
Title was: WebdriverIO · Next-gen WebDriver test framework for Node.js
```

Yay, Congratulations! You've just run your first automation script with WebdriverIO. Let's step it up a notch and create a real test.

Let's get serious

(If you haven't already, navigate back to the project root directory)

This was just a warm up. Let's move forward and run WebdriverIO with the test runner. If you want to use WebdriverIO in your project for integration testing we recommend using the test runner because it comes with a lot of useful features that makes your life easier. With WebdriverIO v5 and up, the testrunner has moved into the `@wdio/cli` NPM package.

Before installing the test runner, we need to initialize an empty NPM project (this will allow us to the cli to install needed dependencies).

To do this, run:

```
$ npm init -y
```

The `-y` will answer 'yes' to all the prompts, giving us a standard NPM project. Feel free to omit the `-y` if you'd like to specify your own project details.

Now we need to install the cli. Do that by running:

```
ɱm i --save-dev @wdio/cli
```

We'll next want to generate a configuration file that stores all of our WebdriverIO settings. To do that just run the configuration utility:

```
$ ./node_modules/.bin/wdio config
```

A question interface pops up. It will help to create the config easy and fast. If you are not sure what to answer follow this answers:

Q: Where should your tests be launched?
A: local

Q: Shall I install the runner plugin for you?
A: Yes

Q: Where is your automation backend located?
A: On my local machine

Q: Which framework do you want to use?
A: mocha

Q: Shall I install the framework adapter for you?
A: Yes (just press enter)

Q: Do you want to run WebdriverIO commands synchronous or asynchronous?
A: sync (just press enter, you can also run commands async and handle promises by yourself but for the sake of simplicity let's run them synchronously)

Q: Where are your test specs located?
A: ./test/specs/*/.js (just press enter)

Q: Which reporter do you want to use?
A: dot (just press space and enter)

Q: Shall I install the reporter library for you?
A: Yes (just press enter)

Q: Do you want to add a service to your test setup?
A: none (just press enter, let's skip this for simplicity)

vel of logging verbosity:
A: trace

Guide API Help Blog

A: http://localhost (just press enter)

That's it! The configurator now installs all required packages for you and creates a config file with the name `wdio.conf.js`. As we're using Geckodriver, we need to override the default path (which uses the Selenium's default of `/wd/hub`). Then, we'll be ready to create your first spec file (test file).

Configure the path

Because we're using geckodriver, the 'path' to it is different from the expected default (which is `/wd/hub`). To change that, we need to edit the `wdio.conf.js` file.

At the top of the file, right after the `exports.config = {` line, add a section to specify the path:

```
    // Override the default path of /wd/hub
    path: '/',
```

So the top of your file should look like:

```
exports.config = {
    // Override the default path of /wd/hub
    path: '/',
    //
    // ====================
    // Runner Configuration
    // ====================
    //
    // WebdriverIO allows it to run your tests in arbitrary locations (e.g. locally or
    // on a remote machine).
    runner: 'local',
    ... the rest of the settings ...
}
```

Be sure to save the file after your changes.

Create Spec Files

Now it's time to create our test file. We're going to store all of our files in a new folder.

```
$ mkdir -p ./test/specs
```

Create a new file in that folder (we'll call it `basic.js`):

```
$ touch ./test/specs/basic.js
```

Open that file up and add the following code to it:

```
const assert = require('assert');

describe('webdriver.io page', () => {
    it('should have the right title', () => {
        browser.url('https://webdriver.io');
        const title = browser.getTitle();
        assert.equal(title, 'WebdriverIO · Next-gen WebDriver test framework for Node.j
    });
});
```

Once added, save, then return to your terminal.

Kick Off Testrunner

The last step is to execute the test runner. To do so just run:

```
$ ./node_modules/.bin/wdio wdio.conf.js
```

Hurray! The test should pass and you can start writing integration tests with WebdriverIO.

If you ran into any issue, reach out in our Gitter Channel and post the error you're seeing, plus the step you're currently on.

If you are interested in more in depth video on-boarding tutorials, feel free to check out our very own course called learn.webdriver.io. Also our community has collected a lot of boilerplate projects that can help you to get started.

I/O

Docs Community More

Guide API Help Blog

API Reference Support Chat GitHub

Help Twitter Star 4,872

JS Foundation