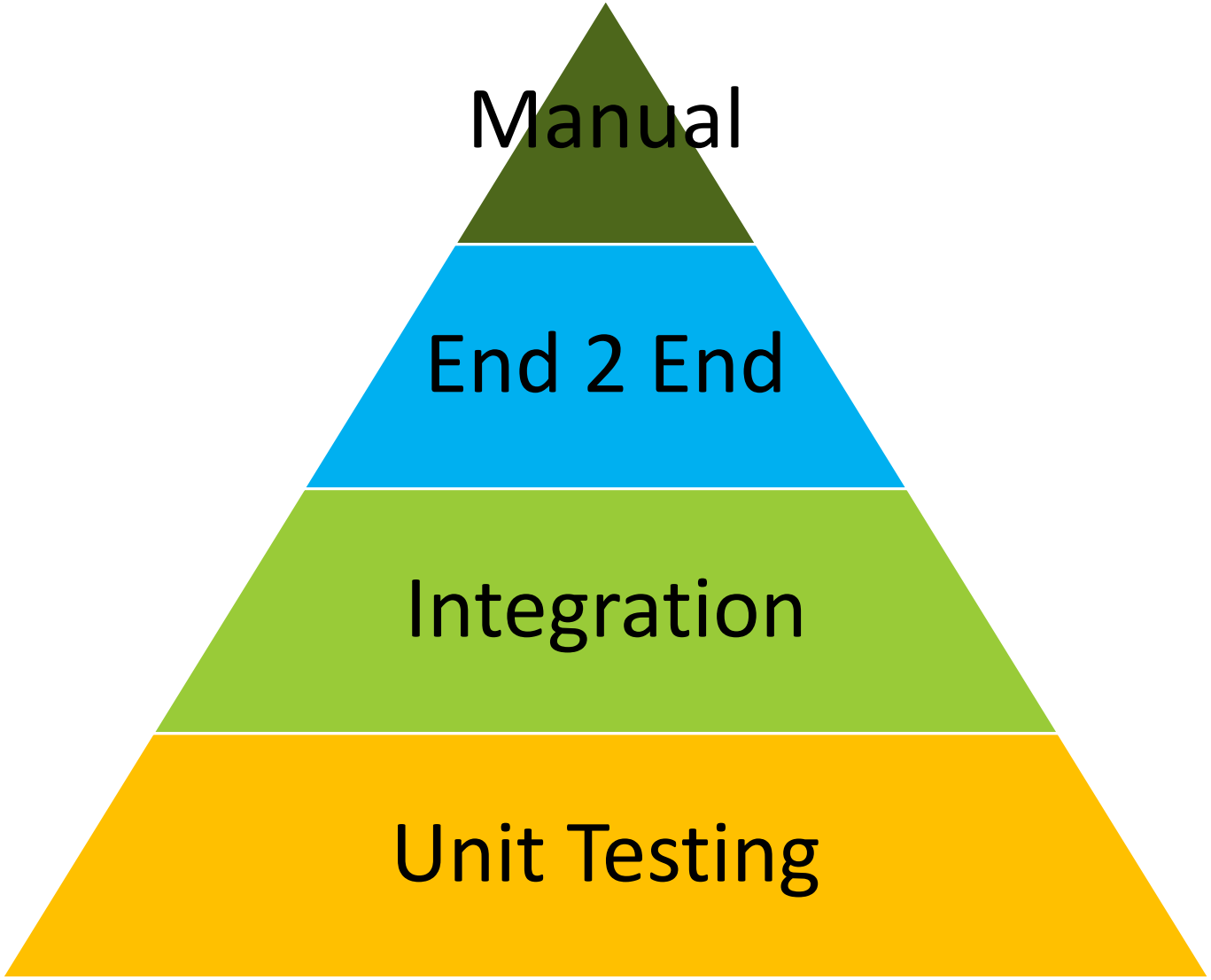
End to End testing with Nightwatch

What are we going to cover

End to End testing with Nightwatch

Testing Triangle



What is end-to-end testing?

"End-to-end testing is a methodology used to test whether the flow of an application is performing as designed from start to finish. The purpose of carrying out end-to-end tests is to identify system dependencies and to ensure that the right information is passed between various system components and systems."

-- Techopedia --

Why write end-to-end tests?

End-to-end tests ensure the **application as a whole** is working as expected

- Unit tests can only verify small units of code

Downside of end-to-end tests

End-to-end tests:

- Run much **slower** than unit tests
- Usually require **more setup** work
- Can be **hard to verify** if a test really passed
- Can be much **harder to debug** when things go wrong

When not to write end-to-end tests

Don’t write end-to-end tests for **algorithms**

- Unit tests are much better for those cases

Don’t write end-to-end tests for **obscure edge cases**

- Manual testing is much better in those cases

Don’t write end-to-end tests for **UI effects** like CSS animations

- Again these are much better to test manually

Nightwatch.js

"Nightwatch.js is an automated testing framework for web applications and websites, written in Node.js and using the W3C WebDriver API (formerly Selenium WebDriver). It is a complete browser (End-to-End) testing solution which aims to simplify the process of setting up Continuous Integration and writing automated tests."

Using Nightwatch.js

Nightwatch is an end-to-end testing framework

- It uses the W3C WebDriver aka Selenium standard
- Requests to the WebDriver are done using an HTTP api

Install Nightwatch **using npm**

- Frequently used with Selenium standalone server
- Can also be used with the Chrome driver directly

Nightwatch configuration

nightwatch.json

```
{
  "src_folders": ["./tests/"],
  "globals_path": "./global.js",
  "test_settings": {
    "default": {
      "selenium_port": 9515,
      "default_path_prefix": "“,
      "launch_url": "http://www.google.com"
    }
  }
}
```

Global.js

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

module.exports = {
  waitForConditionTimeout: 15000,
  before(cb) {
    chromedriver.start([], true)
      .then(() => {
        console.log('Chromedriver is ready');
        cb();
      });
  },
  after(cb) {
    chromedriver.stop();
    cb();
  }
};
```

Sample end to end test

```
module.exports = {
  Search_using_Google: function(browser) {
    browser
      .url(browser.launchUrl)
      .waitForElementVisible('body')
      .assert.title('Google')
      .waitForElementVisible('input[type=text]')
      .setValue('input[type=text]', 'rembrandt van rijn')
      .waitForElementVisible('input[name=btnK]')
      .click('input[name=btnK]')
      .pause(1000)
      .assert.containsText('div#search',
        'Rembrandt van Rijn - Wikipedia')
      .end();
  }
};
```

Running the test

```
[Google Test] Test Suite
==============================

Running:  Search_using_Google
Starting ChromeDriver 2.44.609538 (b655c5a60b0b544917107a59d4153d4bf78e1b90) on port 9515
Only local connections are allowed.

DevTools listening on ws://127.0.0.1:57882/devtools/browser/c46e9755-86c8-48ee-ba9b-d31093848249
 √ Element <body> was visible after 36 milliseconds.
 √ Testing if the page title equals "Google".
 √ Testing if element <input[type=text]> is visible.
  Warn: WaitForElement found 2 elements for selector "input[name=btnK]". Only the first one will
 be checked.
 √ Element <input[name=btnK]> was visible after 92 milliseconds.
 √ Testing if element <div#search> contains text: "Rembrandt van Rijn - Wikipedia".

OK. 5 assertions passed. (5.396s)

[Top Movies Test] Test Suite
==================================
```

Page objects

Using **page object** commands and elements can make tests much **easier to read**

- The page objects abstract the implementation details into functional items

Testing with page objects

```
Google_search_using_page_Objects:
  function (browser) {
    var google = browser.page.google();
    google.navigate()
      .checkTitle()
      .searchFor('rembrandt van rijn')
      .checkForResult(
        'Rembrandt van Rijn - Wikipedia');
    browser.end();
}
```

Part of the page objects used

```
module.exports = {
  url: 'http://www.google.com',
  elements: {
    input: 'input[type=text]', button: 'input[name=btnK]'
  },
  commands: [{
    searchFor: function (value) {
      this
        .waitForElementVisible('@input')
        .setValue('@input', value)
        .waitForElementVisible('@button')
        .click('@button');
      return this;
    }
  }]
}
```

Conclusion

Nightwatch is a great tool for End to End testing

Use the Page Object Model to make test easier to read and maintain