Javascript Testing Frameworks and Tools (Jasmin, Jest, Mocha, Tape, Cypress)

Azat Satklichov

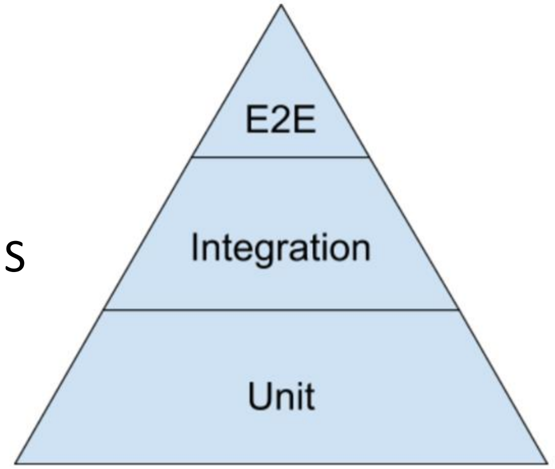
azats@seznam.cz,
http://sahet.net/htm/java.html,
https://github.com/azatsatklichov/java-and-ts-tests

Agenda

- ❑ Types of Javascript Testing Frameworks & Tools
- ❑ Most Used Testing Tools
- ❑ Functional Testing Tools
- ❑ Jasmin
- ❑ Jest
- ❑ Mocha + Chai + Sinon
- ❑ Choose Your Unit and Integration Tests Framework
- ❑ Tape
- ❑ Choose Your Functional Tests (AAT) Framework
- ❑ Cypress

Javascript Testing via Java

- **Rhino** is a JavaScript engine written fully in Java and managed by the Mozilla Foundation, started at Netscape in 1997
- 2011 **Nashorn** is a JavaScript engine, 2014 part of Java 8 (Rhino in Java7 replaced)
- 2018, Java 11, Nashorn is deprecated, and has been removed from JDK 15 onwards



JS Tests, Testing Frameworks and Testing Tools

Types of Tests: Unit Tests, Integration Tests, E2E Tests **Running:** Browser, Headless, NodeJS

Test launchers(runners): Karma, Jasmine, Jest, TestCafe, Cypress, webdriverio
Testing structure providers: **Mocha**, **Jasmine**, **Jest**, Cucumber, TestCafe, Cypress,
Assertion functions: Chai, **Jasmine**, **Jest**, Unexpected, TestCafe, Cypress, Assert.js, Should.js
Mocks, spies, and stubs: Sinon, **Jasmine**, enzyme, **Jest**, testdouble
Generate and compare snapshots: **Jest**, Ava
Generate code coverage: Istanbul, **Jest**, Blanket
Browser Controllers (crawl, structure, screenshot, form-sub.): Nightwatch, Nightmare, Phantom**,** Puppeteer, TestCafe, Cypress
Visual Regression Tools: Applitools, Percy, Wraith, WebdriverCSS, Hapo, LooksSame, BackstopJS, AyeSpy, reg-suit, Differencify
Functional Testing Tools (Automated Acceptance Testing): Selenium WebDriver, Protractor, WebdriverIO, Nightwatch, Appium, TestCase, Cypress, Puppeteer, **Playwright**, PhantomJS, Nightmare, CodeceptJS
No Coding Functional Testing Tools: testim, Chromatic, Screener, Ghost Inspector

E.g. Some frameworks e.g. Jest, Jasmine (still needs CC), TestCafe, and Cypress provide all of these out of the box.
Some provides only spec. functionality, then **combinations of tools** would be used: mocha + chai + sinon.

Most Used Testing Tools

- **jsdom** simulated browser env., tests run fast. But not everything can be simulated, e.g. can't take a screenshot..

- **Testing Library** testing utilities encourages good testing practices, helps to test UI components in a user-centric way

- **Istanbul / NY (->, es6)** tells how much of your code is covered with unit tests. **Jest/Tap** has by default Istanbul

- **Karma** hosts a test server with a special web page to run your tests in the page's environment.

- **Chai** is the most popular assertion library. It has many plugins and extensions.
- **Unexpected** is an assertion library with a slightly different syntax from Chai.

- **Enzyme** is used to render components and makes it easier to test your React Components
- **Sinon** has powerful standalone test spies, stubs and mocks for JS to work with any framework (Mocha, Tape,..).
- **testdouble** like Sinon but with better in design, philosophy, and features that could make it useful in many cases.

- **Wallaby** runs on your IDE (e.g. IntelliJ, MOCHA SIDEBAR) and runs tests, shows anything fails in real time alongside your code.
- **Cucumber** help with writing tests in BDD by dividing them between the acceptance criteria files using the **Gherkin** syntax and the tests that correspond to them.

Functional Testing Tools

Selenium WebDriver dominated the market of Functional Tests for years. ..

Protractor wraps Selenium and provides us with improved syntax and special built-in hooks for **Angular.**

WebdriverIO has its own implementation of the selenium WebDriver.

Nightwatch has its own implementation of the selenium WebDriver.

Apium provides an API similar to Selenium for testing websites on a mobile devices iOS, Android, Windows Phone

TestCafe is a great alternative to Selenium-Based tools. It was rewritten and **open-sourced** at the end of 2016.

Cypress is a direct competitor of TestCafe. Cypress.io runs itself in the browser and controls your tests from there where TestCafe runs in Node.js and controls the tests through a serialized communication with its injected script in the browser.

Puppeteer developed by **Google**. It provides a convenient Node.js API to control Chrome or **Headless Chrome.**

Playwright is a exactly like **Puppeteer**, but it is developed by **Microsoft**

PhantomJS implements the chromium engine to create a controllable Chrome-like headless browser.

Nightmare offers a very simple test syntax. Uses Electron which uses Chromium to control the browser's behavior.

Codecept like CucumberJS it provides another abstraction, different philosophy that focuses on user behavior.

Jasmine

Jasmine is a behavior-driven development (BDD) framework for testing JavaScript code. It does not depend on any other JavaScript frameworks. Can be used to write tests for React apps. as well.

Why Use Jasmine?

- ❖ Jasmine does not depend on any other JavaScript framework.
- ❖ Jasmine does not require any DOM.
- ❖ All the syntax used in Jasmine framework is clean and obvious so that you can easily write tests.
- ❖ Jasmine is heavily influenced by Rspec (BDD testing for Ruby), JS Spec, and JSpec (Java test assertions).
- ❖ Jasmine is an open-source framework, versions available like stand-alone, ruby gem, Node.js, etc.

- ❖ **Ready-To-Go:** Comes with everything you need to start testing.
- ❖ **Globals:** Comes with all the important testing features in the global scope as well.
- ❖ **Community:** It has been on the market since 2009 and gathered a vast amount of articles, suggestions and tools that are based on it.
- ❖ **Angular:** Has widespread Angular support and it is recommended in the official Angular documentation.

Jasmine API

`Jasmine Matchers:` **Inbuilt** matchers (toEqual, toBe, not..., toBeTruthy, toThrow, ...) and **Custom** matchers – addMatchers

`Setup and Teardown:` Jasmine provides the global beforeEach, afterEach, beforeAll, and afterAll functions.

Another way to share variables between a beforeEach, **it,** and afterEach is through the **this** keyword.

`xdescribe, xit:` Suites, blocks can be disabled(skipped) via xdescribe, xit functions respectively. Pending specs do not run, but shown

`Spies:` Jasmine has test double functions called spies. spyOn, createSpy, createSpyonObj,
Special matchers to interacting with spies: toHaveBeenCalled, toHaveBeenCalledTimes, toHaveBeenCalledWith

`Matching with more finesse:` Jasmine .any, .anything, .objectContaining, .arrayContaining, .stringMatching

`Custom asymmetric equality tester:` custom asymmetric equality providing an object that has asymmetricMatch function.

`Jasmine Clock, mocking Date:` `jasmine.clock() .install(), .uninstall(), .tick(), .mockDate()`

Jest (lsp-extension FE uses it)

- **Jest** (based on Jasmine) is the **testing framework** created and maintained by **Facebook**. Self sufficient.
- **Performance -** faster for big projects with many test files by implementing a **parallel testing mechanism**.
- **Ready-To-Go** has assertions, spies, and mocks, no need combination-of-tools like mocha + chai + sinon
- **Globals** as in Jasmine, can be considered bad, it makes your tests less flexible but makes your life easier
- **Snapshot testing -** is to ensure that your app's UI doesn't unexpectedly change between releases.
- **Great modules mocking** - Easy way to mock heavy modules to improve testing speed.
- **Code coverage** - Includes a powerful and fast built-in code coverage tool that is based on Istanbul.
- **Reliability**- Has a huge community, used in many very complex projects
- **Support-** It is currently supported by all the major IDEs and tools.
- **Development-** jest only updates the files updated, so tests are running very fast in watch mode.

Mocha + Chai + Sinon

Mocha is the most used library.
Unlike Jasmine, it is used with third party assertions, mocking, and spying tools (usually Sinon and Chai).

- ❖ Community- Has many plugins and extension to test unique scenarios.

- ❖ Mocha includes the test structure as globals, saving time by not having to include or require it in every file.

- ❖ Mocha is **a little** harder to set up and divided into more libraries but it is more flexible and open to extensions.

- ❖ Flexibility in it’s assertions, spies and mocks is highly beneficial.

Choose Your Unit and Integration Tests Framework

The first choice you should probably make is which framework you want to use.

- *Angular apps first choice is* ***Jasmin***. Clean and obvious so that you can easily write tests.
- ***Jest*** ***(****Jasmin based) is very fast, clear, has many features in case you need to cover complex scenarios.*
- *If you want a very flexible and extendable configuration, go with* ***Mocha*** ***(Mocha+Chai+Sinon)****.*
- *If you are* ***minimalist*** *go with* ***Ava****. (no globals, install libs for more: mock, snapshot, parallelism)*
- ***QUnit*** *is mainly to test the jQuery (core, UI, and Mobile) JS libs but can be used to test other JS apps.*

What's Wrong with Mocha, Jasmine, etc…?

- *1. A lot config.(runner, assertion, report lib, ..). 2. Globals (`describe`, `it`, before ..) 3. Shared State(before..)*
  Moreover: Above tools leads to analysis paralysis (wide API tries e2e solution, code smell - non used mocks, …)

Tape

- tape *is a modular testing library* . Simplify your tests and app. by breaking into more modular chunks.
  1. Just loaded. 2. Simple module export. 3. Instead, call setup and teardown, & contain state to local test var
- If you prefer low-level, or dev. choice – writing maintainable tests) [*plan, deepEqual, looseEquals, equals, .. blue-tape]*
- Mock services - proxyquire module makes the process quite easy, require('proxyquire'), require('sinon');
- TAP-producing (tap-dot,nyc..) test harness for node & browsers. Its API is a small superset of the node core assert module.

Choose Your Functional Tests (AAT) Framework

Tools for the purpose of **functional testing** differ very much from each other in their implementation, philosophy, and API. So better understand the different solutions and testing them on your product.

- ❖ *If you want to "just get started" with a simple to set-up cross-browser all-in-one tool, go with* ***TestCafe****.*
- ❖ *For a convenient UI, clear doc., overall fun all-in-one tool Functional Testing experience go with* ***Cypress.io****.*
- ❖ *If you prefer older and more time-proven tools(*), you can "just get started" with* ***Nightwatch.js****.*
- ❖ *(*) with the maximum community support and flexibility, Selenium* ***Webdriver/IO*** *is the way to go.*
- ❖ *If you want the most reliable and Angular friendly solution, use* ***Protractor****.*
- ❖ *New, open-source, JavaScript-based, cross-browser automation library (aims fast&reliable) for E2E* **Playwright**

Automated Acceptance Testing Frameworks (other Langs): FitNesse (Java, ..), Robot[RIDE] (Python), etc.

A set of tools are built on top of Selenium to make this process even faster by directly transforming the BDD specifications into executable code. Some of these are ***JBehave, Capybara and Robot Framework***.

Cypress

Cypress is a free and open source automation tool, MIT-licensed and written in JS.

- **Parallel testing** was introduced in version 3.10.
- **Documentation**- Solid and clear.
- **Native access to all your application's variables** without serialization (TestCafe on the other hand turns objects to JSON, sends them to Node.js as text and then parses them back to objects).
- **Very convenient running and debugging tools**- Easy debugging and logging of the test process.
- **Cross-browser Support**- since version 4.0.
- **Some use-cases are missing** but in constant development such as lack of HTML5 drag-n-drop.
- **Using Mocha** as the test structure provider makes its use pretty standard and allow your functional tests to be built in the same structure as the rest of your tests.

THANK YOU

References

https://github.com/azatsatklichov/java-and-ts-tests
https://medium.com/welldone-software/an-overview-of-javascript-testing-7ce7298b9870
https://catonmat.net/writing-javascript-tests-with-tape
https://ci.testling.com/
https://codecept.io/basics/
https://blog.logrocket.com/unit-testing-node-js-applications-using-mocha-chai-and-sinon