Research Frameworks

This document contains the results of the Frameworks research.
While researching, the results of other researches (that were performed within the same sprint) were taken into account.
The result of the IDE research was the most important one for this research. Since we're using WebStorm, any framework that we use should be easily integratable with the IDE.

Javascript Framework Suggestions

jQuery – 'Write less, do more.'

https://jquery.com/
jQuery is a fast, small, and feature-rich JavaScript library. It makes things like HTML document traversal and manipulation, event handling, animation, and Ajax much simpler with an easy-to-use API that works across a multitude of browsers. With a combination of versatility and extensibility, jQuery has changed the way that millions of people write JavaScript.
+ Easily used by millions of people
+ Just JavaScript, only complicated actions (like AJAX calls or attaching handlers) have been abstracted
+ We already know how to use it

AngularJS – 'Superheroic JavaScript MVW Framework'

https://angularjs.org/
HTML is great for declaring static documents, but it falters when we try to use it for declaring dynamic views in web-applications. AngularJS lets you extend HTML vocabulary for your application. The resulting environment is extraordinarily expressive, readable, and quick to develop.
+ Easy to use
- Is an entire MVC framework, while we only need a controller framework

React.js – 'A JavaScript library for building user interfaces'

http://facebook.github.io/react
React abstracts away the DOM from you, giving a simpler programming model and better performance. Since React makes no assumptions about the rest of your technology stack, it's easy to try it out on a small feature in an existing project.
+ Virtual DOM
+ Performance
- Is a bit overkill since we do not focus on designing a UI

Javascript Framework Recommendation

jQuery, as our project does not focus on designing a UI and we just need a framework that makes common actions like AJAX calls easier to perform. Another great advantage is that we have all used it before in the course Web- and Database AngularJS is more powerful than jQuery, but there is no need for us to transform the entire DOM while we have only very little HTML. The same goes for React.js.

Type Annotation Framework Suggestions

Flow – ‘A static type checker for JavaScript’

http://flowtype.org/
Flow is a static type checker, designed to quickly find errors in JavaScript applications. Typed JavaScript code with Flow annotations easily transforms down to regular JavaScript, so it runs anywhere.
+ Relies heavily on type inference
+ Because of that, easily mix dynamic code with static code
+ Can catch errors related to `null`
- Does not work on Windows

TypeScript – ‘JavaScript that scales’

https://www.typescriptlang.org/
TypeScript is a typed superscript of JavaScript that compiles to plain JavaScript.
+ Types are optional, but can be used for static checking
+ Support for type-checking virtually any JavaScript library on the web
- Building options are not very versatile (only compile to same directory or link everything in one file)

Babel TypeCheck

https://github.com/codemix/babel-plugin-typecheck
This is a Babel plugin for static and runtime type checking using flow type annotations. This plugin converts javascript with type annotations to plain javascript with run-time type checks.
+ Run-time support for types
+ Supports primitive types, like int8 or uint64 or the like
- Less support for integration

Test Runner Framework Recommendation

TypeScript, as this framework has the best support in WebStorm. Flow seems just a little bit better than TypeScript, but as all of our developers work with Windows, this is unfortunately not an option.

Test Runner Framework Suggestions

Karma – ‘A simple tool that allows to execute JavaScript in multiple *real* browsers.’

https://github.com/karma-runner/karma
Karma is not a testing framework, nor an assertion library. Karma just launches a HTTP server, and generates the test runner HTML file you probably already know from your favourite testing framework.
+ Supports any testing framework
+ Supports running on any browser (though we only use Chrome)
+ Runs the test on every save

Mocha – ‘simple, flexible, fun’

https://mochajs.org/
Mocha is a feature-rich JavaScript test framework running on Node.js and in the browser, making asynchronous testing simple and fun. Mocha tests run serially, allowing for flexible and accurate reporting, while mapping uncaught exceptions to the correct test cases.
+ Includes a testing framework for unit and end-to-end testing
+ Requires little configuration; automatically looks for tests inside a `test/` folder
- Little integration in WebStorm

Wallaby.js – ‘wallaby.js runs your code as you write it’

https://wallabyjs.com/
Wallaby.js is an intelligent and super fast test runner for JavaScript that continuously runs your tests. It reports code coverage and other results directly to your code editor, immediately as you change your code. The tool provides a huge productivity boost whether you are doing TDD/BDD or using any other approach.
+ Insanely fast, because it only executes tests affected by your code changes and runs your tests in parallel.
+ Used by many huge companies
- Insanely expensive ($100 for a personal licence)

Test Runner Framework Recommendation

Karma, as we decided we want a good integration in WebStorm. Mocha provides a lot of extra functionality for testing, but we can also provide these with other frameworks. Wallaby has a really good reputation, the only thing that makes it unusable to us students is the expensive licence.

JavaScript Unit Test Framework Suggestions

QUnit – ‘A JavaScript Unit Testing Framework’

http://qunitjs.com/
QUnit is a powerful, easy-to-use JavaScript unit testing framework. It's used by the jQuery, jQuery UI and jQuery Mobile projects and is capable of testing any generic JavaScript code, including itself!
+ Based on the CommonJS Unit Testing standard
+ Simple
- A bit too simple
- Hardly anybody uses this, i.e. little support online

Jasmine – ‘A JavaScript Testing Framework’

http://jasmine.github.io/
Jasmine is a behavior-driven development framework for testing your JavaScript code. Its syntax looks a lot like the Ruby on Rails testing framework, and is behaviour-driven.
+ It does not depend on any other JavaScript frameworks
+ It does not require a DOM
+ Behaviour-driven, but can also be used for test-driven (which are nearly the same thing)
+ Most people use this, i.e. more support online
- Less support for virtual server calls (but it is possible)

Mocha – ‘simple, flexible, fun’

https://mochajs.org/
Mocha is a feature-rich JavaScript test framework running on Node.js and in the browser, making asynchronous testing simple and fun. Mocha tests run serially, allowing for flexible and accurate reporting, while mapping uncaught exceptions to the correct test cases.
+ Works on unit level and on browser level
+ Support for ‘before’ and ‘after’ functions, like in JUnit for Java
- Little integration in WebStorm

JavaScript Unit Test Framework Recommendation

Jasmine, as we really want the WebStorm integration. QUnit is too unused and too simple for our taste, it doesn’t meet our needs. Jasmine and Mocha are mostly alike in functionality, but as we already disliked Mocha as test runner for the little integration in WebStorm, Jasmine wins this one.

JavaScript End-to-End Test Framework Suggestions

Protractor – ‘end to end testing for AngularJS’

http://angular.github.io/protractor
Protractor is an end-to-end test framework for AngularJS applications. Protractor runs tests against your application running in a real browser, interacting with it as a user would.
+ Easy configuration
+ Automatically waits/sleeps during tests
- Heavily depends on the AngularJS framework, while we are not using that.
- Uses a Selenium server, which is an additional dependency

Nightwatch.js – ‘Browser automated testing done easy.’

http://nightwatchjs.org/
Nightwatch.js is an easy to use Node.js based End-to-End (E2E) testing solution for browser based apps and websites. It uses the powerful Selenium WebDriver API to perform commands and assertions on DOM elements.
+ Clean syntax
+ Easy to use by providing CSS selectors and BDD
- Uses a Selenium server, which is an additional dependency

WebDriverIO – ‘Selenium 2.0 bindings for NodeJS’

http://webdriver.io/
WebdriverIO lets you control a browser or a mobile application with just a few lines of code. Your test code will look simple, concise and easy to read. The integrated testrunner allows you to write asynchronous commands in a synchronous way so that you don’t need to care about how to propagate a Promise to avoid racing conditions.
+ Very easy to set up and use
+ Can work with the most used Unit Test frameworks, including Jasmine
± Works with Travis, though a bit cumbersome
- Uses a Selenium server, which is an additional dependency

JavaScript End-to-End Test Framework Recommendation

WebDriverIO. Protractor looked really nice at first, except we don’t use AngularJS. Nightwatch and WebDriverIO both look very promising, but they also share a pitfall, which is having to use an additional dependency, a Selenium server. The nice thing about WebDriverIO is that it can be used with our current Unit test framework, only this time the behaviour driven design part can be used, and this is the reason why we chose it in the end.