Test Report

Sprint 8

This document elaborates on the status of the source code in terms of testing. Reasons are added here only for why classes don't have 100% coverage. All classes that aren't included to the discussion, are considered as fully covered.
In addition to this, we'd like to suggest the reader to analyze the coverage report as well, that can be found in the coverage folder (after having performed the necessary steps, described in the README.md). Consider this document as a supplement to the coverage report.
The Coveralls coverage changes can be found here: Coveralls.
At the end of the report, an overview has been added.

The test coverage of the source code is 94% at the end of sprint 8. This amount is close to the maximum percentage that we can reach. We have documented which classes we can not (fully) test in a list below.

List of not (fully) covered classes

The Controllers

- The ContentController has been covered for 29%.
  This has to do with the fact that we can't create unit tests for this, as it would require us to interact with the DOM, which would actually make it an integration test. Besides, the functions that are called are private, so we have to reach these methods by getting from the public start methods, to the private methods, while applying a lot of spies. The tests would become unclear and what we're testing is calling a lot of different classes.
- The MainController has been covered for 43%.
  The same reasons apply here as for the ContentController.
  Besides, a lot of functions are used that involve chrome in which new functions are created. The functions are not covered as we can't reach them via the ChromeMock class (the ChromeMock class mocks the functions, so they don't cover any implementation). It would require us to change the MainController methods and extract the functions that are used in the used chrome functions and test these.

DoNotWatchOptions

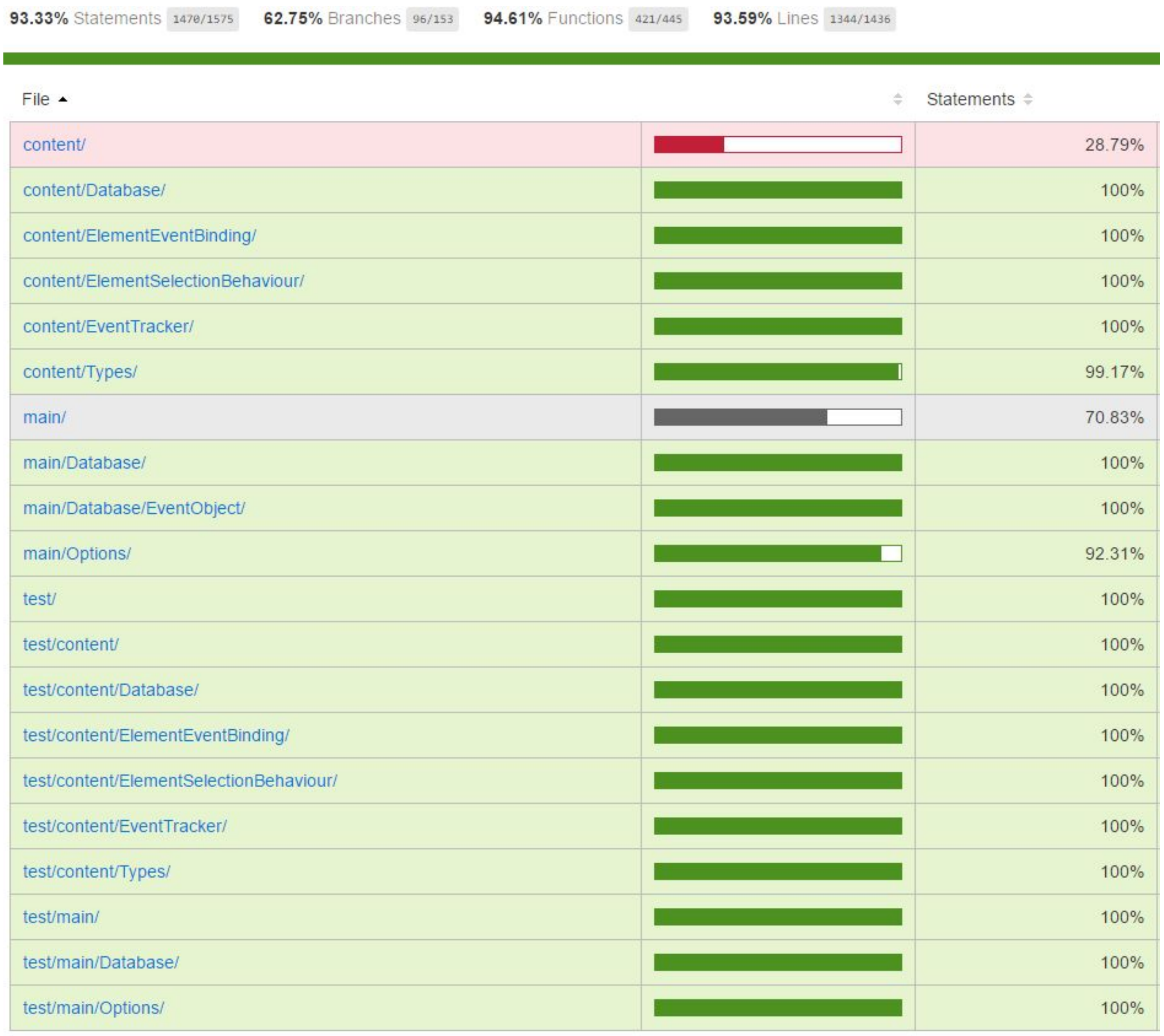
DoNotWatchOptions has been covered for 91%.
The getCombinations function has not yet been covered, as the function is still a placeholder for something that still has to be implemented. Thus it can't yet be covered.

Options

Options has been covered for 100% in line coverage, but 91.3% in function coverage and 58.33% in branch coverage.
Because a few small lines that
Options has a few small unreachable lines and uncoverable branches.
The few unreachable lines involve internal lines used in chrome functions. We can’t reach these, the ChromeMock mocks these functions, so they have no functionality.
The few uncoverable branches involve the use of the hasOwnProperty function inside an if-check, which is a good checking habit when using a for-each loops in TypeScript, but can’t be made false in a test.

Overview

The following overview is extracted from the coverage report at the end of sprint 7. The percentages represent the line coverage per item.



93.33% Statements 1470/1575 **62.75%** Branches 96/153 **94.61%** Functions 421/445 **93.59%** Lines 1344/1436

File ▲		Statements
content/		28.79%
content/Database/		100%
content/ElementEventBinding/		100%
content/ElementSelectionBehaviour/		100%
content/EventTracker/		100%
content/Types/		99.17%
main/		70.83%
main/Database/		100%
main/Database/EventObject/		100%
main/Options/		92.31%
test/		100%
test/content/		100%
test/content/Database/		100%
test/content/ElementEventBinding/		100%
test/content/ElementSelectionBehaviour/		100%
test/content/EventTracker/		100%
test/content/Types/		100%
test/main/		100%
test/main/Database/		100%
test/main/Options/		100%