Test plan

Simon Jonsson
sjovl08@student.lnu.se
https://github.com/sjovl/sjovl08_1dv600

Objective

The objective is to test the implementations to the last iteration.

What to test and how

I intend to test UC1 and UC2 by writing manual test cases.
I will also write automated unit tests for the Hangman methods in the class HangmanTest.

Time plan

Task	Estimated	Actual
Manual TC	2h	1h
Unit tests	2h	3h
Running manual tests	30min	20min
Code inspection	1h	1h
Test report	1h	1h

Manual tests

TC1.1 Start game

Use case: UC1 Start game.
Scenario: Game menu is displayed.

The main scenario of UC1 is tested when the app is started.

Precondition: None

Tests steps
Start the app

Expected
The system displays the game menu

TC2.1 Set nickname

Use case: UC2 Set nickname.
Scenario: Nickname is stored successfully.

The main scenario is tested where a user stores their nickname successfully.

Precondition: The game is running.

Test steps

Choose set nickname in menu.
System shows input nickname.
Enter nickname “Simon” and press enter.

Expected
The system should show the game menu.

Test report

Test	**UC1**	**UC2**	**UC3**	**UC4**
TC1.1	1/OK			
TC2.1		1/OK		
Coverage & success	1/OK	1/OK		

Test	**Hangman**
HangmanTest	31%/FAIL
Coverage & success	31%/FAIL

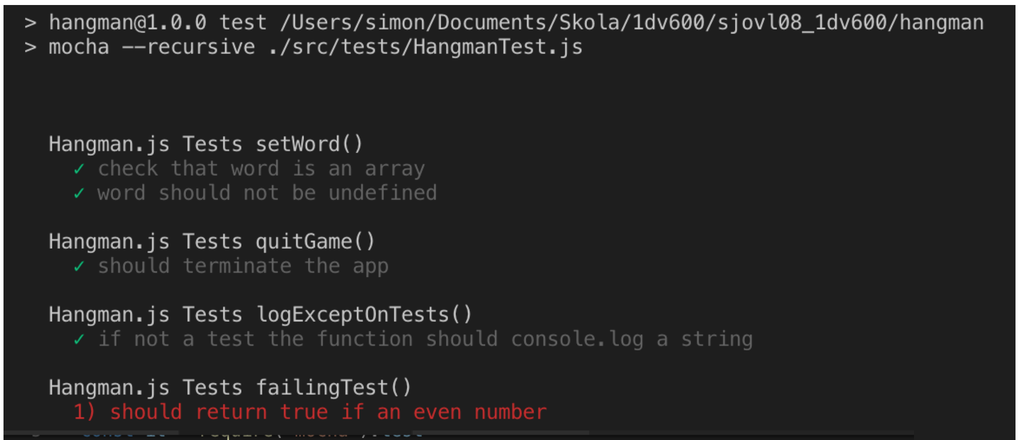
Comment

80% of tests OK. 20% fail.
Had a hard time figuring out what to test in some methods.
Changed some code to get return values to test.

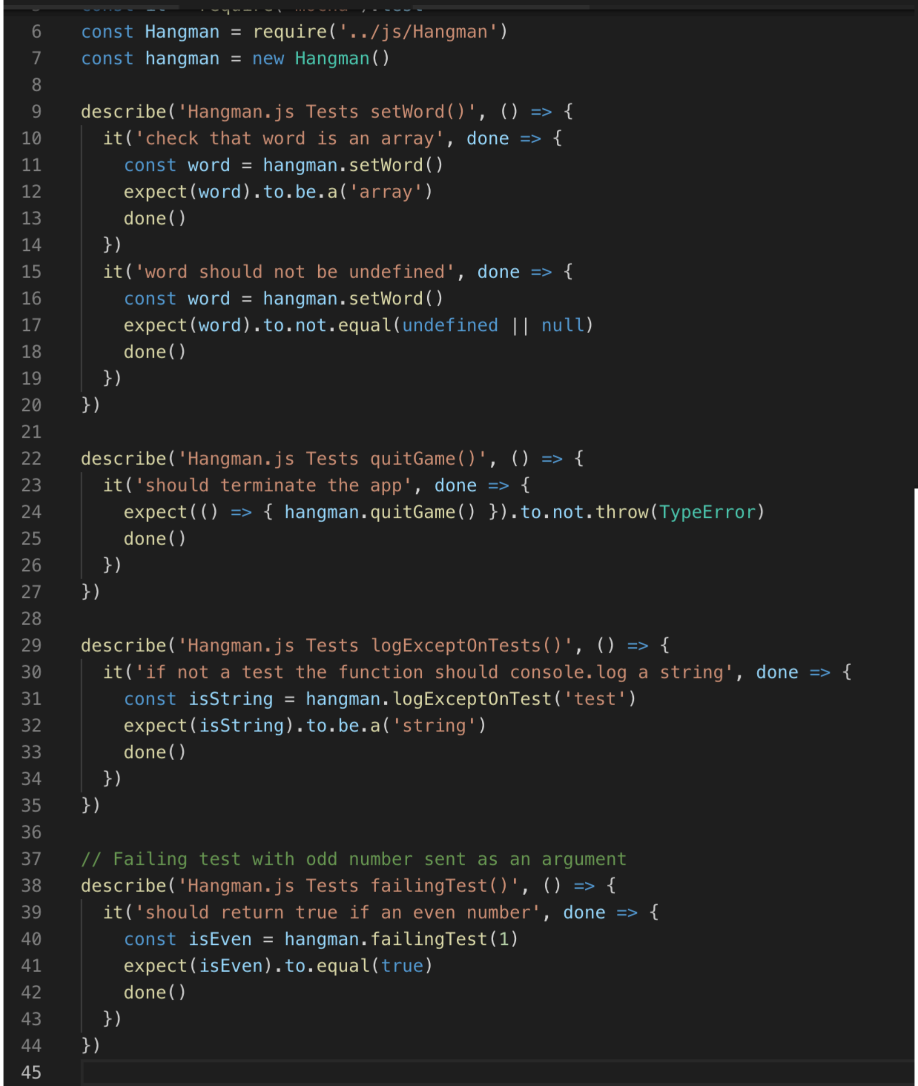
Reflection

Since I’ve never used any testing code before I had to spend some time to learn mocha and chai to be able to write some good looking test code.
When I got used to it I found it quite useful to be able to test my code and it is something I definitely will use in the future projects even before I write my code to be able to start of correct in my methods from the start.
I had some troubles figuring out what exactly to do with the testing and did not expect it to be this requiring.

Test screenshots



```
> hangman@1.0.0 test /Users/simon/Documents/Skola/1dv600/sjovl08_1dv600/hangman
> mocha --recursive ./src/tests/HangmanTest.js



  Hangman.js Tests setWord()
    ✓ check that word is an array
    ✓ word should not be undefined

  Hangman.js Tests quitGame()
    ✓ should terminate the app

  Hangman.js Tests logExceptOnTests()
    ✓ if not a test the function should console.log a string

  Hangman.js Tests failingTest()
    1) should return true if an even number
```



```javascript
const Hangman = require('../js/Hangman')
const hangman = new Hangman()

describe('Hangman.js Tests setWord()', () => {
  it('check that word is an array', done => {
    const word = hangman.setWord()
    expect(word).to.be.a('array')
    done()
  })
  it('word should not be undefined', done => {
    const word = hangman.setWord()
    expect(word).to.not.equal(undefined || null)
    done()
  })
})

describe('Hangman.js Tests quitGame()', () => {
  it('should terminate the app', done => {
    expect(() => { hangman.quitGame() }).to.not.throw(TypeError)
    done()
  })
})

describe('Hangman.js Tests logExceptOnTests()', () => {
  it('if not a test the function should console.log a string', done => {
    const isString = hangman.logExceptOnTest('test')
    expect(isString).to.be.a('string')
    done()
  })
})

// Failing test with odd number sent as an argument
describe('Hangman.js Tests failingTest()', () => {
  it('should return true if an even number', done => {
    const isEven = hangman.failingTest(1)
    expect(isEven).to.equal(true)
    done()
  })
})

```