Moodle Developer meeting - January 2025

Unit tests in JavaScript for plugin development

Stephan Robotta

Bern University of Applied Sciences (BFH)

Motivation

- I am maintainer of several TinyMCE Plugins.
- End-to-End Test covered via Behat.
- Minimal test coverage.

Current Behat tests allow to select down to a paragraph in the TinyMCE content only. They cover mostly all of the menu items.

Missing: click on toolbar buttons, custom selections in the text, context menu and double click on selection.

JavaScript Testframeworks

Where PHP has one widely accepted standard (PHPUnit) JavaScript has a lot of packages:

1. MochaJS
2. Jest
3. Jasmine
4. Karma
5. Puppeteer (Node Library)
6. NightwatchJS
7. Cypress
8. Playwright (Node Library)
9. Selenium

https://www.browserstack.com/guide/top-javascript-testing-frameworks

First pick: Jest

Plugin: Tiny Multi-Language Content (v2)

https://github.com/bfh/moodle-tiny_multilang2

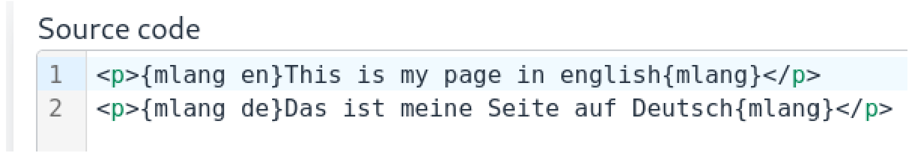
JS Framework: Jest (https://jestjs.io/)

Node.js must be installed!

Note: Whenever ▯ appears in a code block, that means the line break is in the slides only, to fit the content. In the original file the content from the next line would continue at the line where ▯ is located at.

Usecase HTML parser

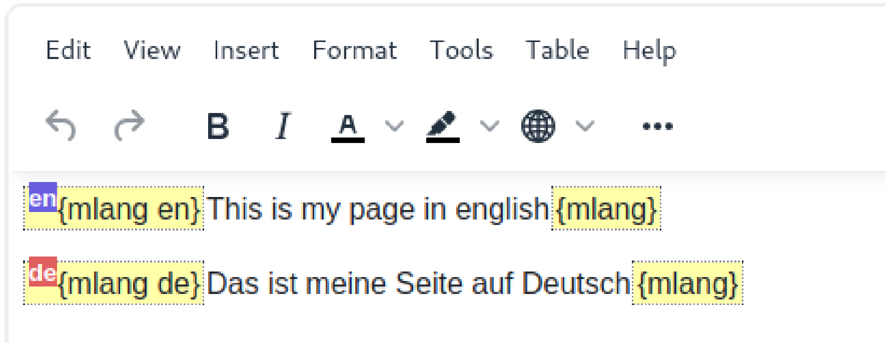
1: Text in Moodle



Source code

```
<p>{mlang en}This is my page in english{mlang}</p>
<p>{mlang de}Das ist meine Seite auf Deutsch{mlang}</p>
```

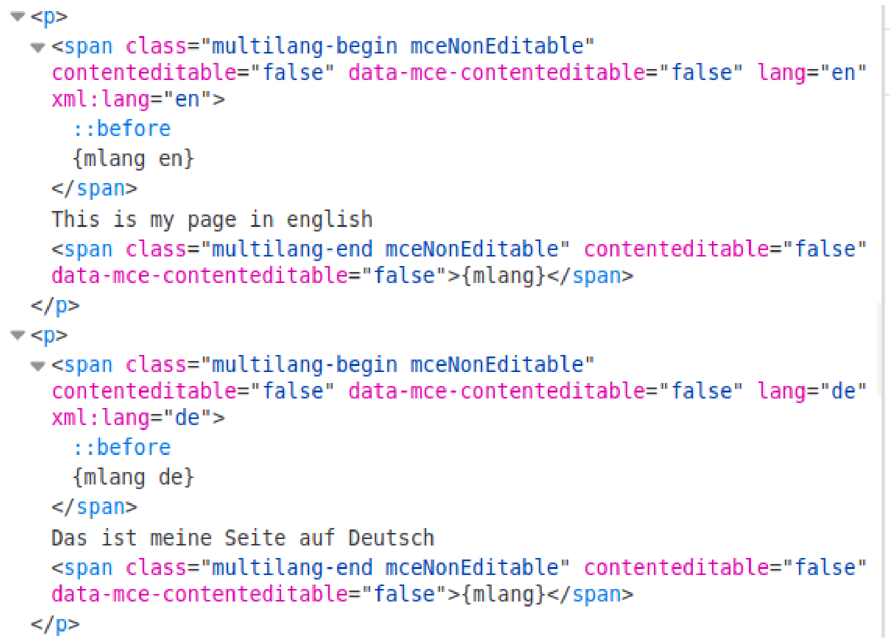
2: Visible in TinyMCE



en{mlang en}This is my page in english{mlang}

de{mlang de}Das ist meine Seite auf Deutsch{mlang}

3: TinyMCE HTML for layout



```
<p>
  <span class="multilang-begin mceNonEditable"
  contenteditable="false" data-mce-contenteditable="false" lang="en"
  xml:lang="en">
    ::before
    {mlang en}
  </span>
  This is my page in english
  <span class="multilang-end mceNonEditable" contenteditable="false"
  data-mce-contenteditable="false">{mlang}</span>
</p>
<p>
  <span class="multilang-begin mceNonEditable"
  contenteditable="false" data-mce-contenteditable="false" lang="de"
  xml:lang="de">
    ::before
    {mlang de}
  </span>
  Das ist meine Seite auf Deutsch
  <span class="multilang-end mceNonEditable" contenteditable="false"
  data-mce-contenteditable="false">{mlang}</span>
</p>
```

Install jest

Inside a js testfolder create a `package.json`:

```
{
    "type": "module",
    "devDependencies": {
        "jest": "^29.7.0"
    },
   "jest": {
        "verbose": true,
        "testMatch": [
            "**/*.test.js"
        ],
        "transform": {}
    }
}
```

Install the package with: `npm install`

CommonJS vs. ES modules

In Moodle ESM modules are used. These are imported not required. The differences are well explained here:
https://dev.to/iggredible/what-the-heck-are-cjs-amd-umd-and-esm-ikm

I want to test the files in `amd/src/*.js`. In order that Jest picks them they need to end with `.mjs` to indicate a ESM module.

Therefore, I copy e.g. `amd/src/htmlparser.js`
to `tests/js/src/htmlparser.mjs` so that via the test case the file can be imported to test it's functions.

JS Unit test

Create a test case file:

```
import {expect, test} from '@jest/globals';
import {parseEditorContent} from './src/htmlparser.mjs';

test('Description what I want to test.', () => {
    const html = '<p>html to parse</p>';
    const parsed = '<p>pared html</p>';
    expect(parseEditorContent(html)).toEqual(parsed);
});
```

Run the test

Change into the plugins `tests/js` directory.

Run: `node --experimental-vm-modules ./node_modules/.bin/jest`

Test can be run on the dev machine directly (where npm is installed). This also works in the plugin repo without having Moodle around it.

In the docker containers (using the official Moodle Docker repo) there is no node installed, so tests cannot be launched inside a container.

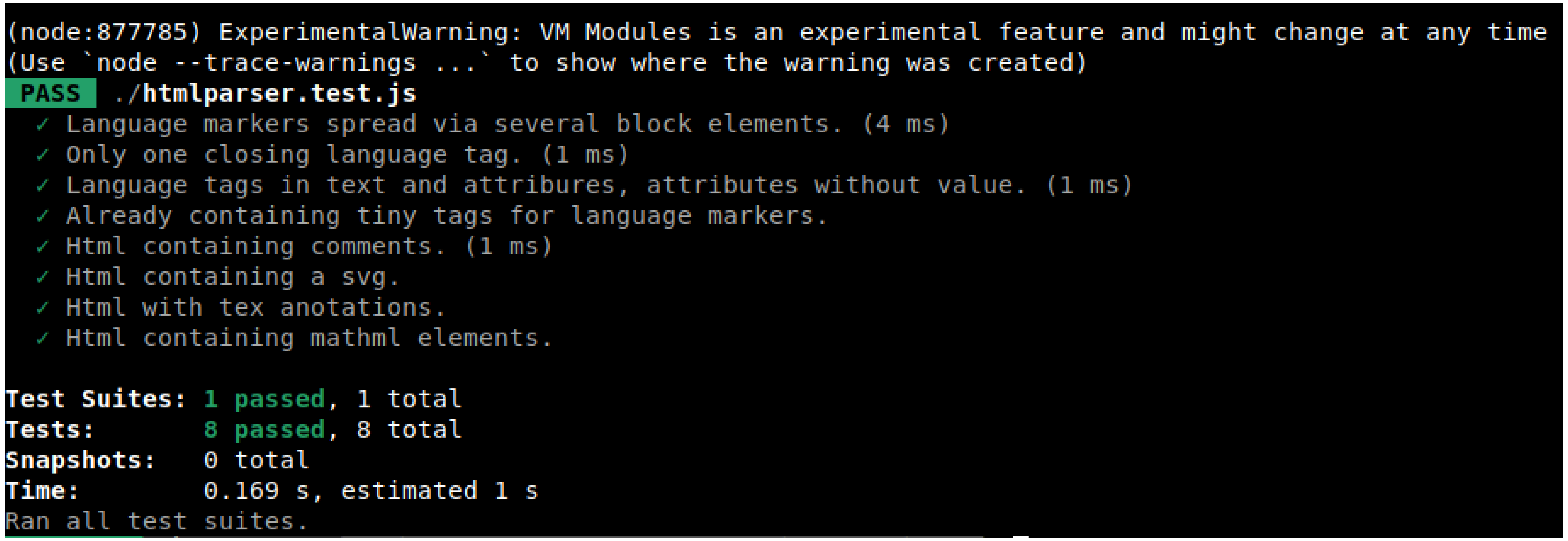
Add npm scripts

Extend the `package.json` with the scripts:

```json
{
    "type": "module",
    "devDependencies": {
        "jest": "^29.7.0"
    },
    "scripts": {
        "cpsrc": "mkdir -p src && for i in ../../amd/src/*.js ; do t=${i##*/}; x=${t%*.js}; cp $i src/${x}.mjs; done",
        "jest": "node --experimental-vm-modules ./node_modules/.bin/jest",
        "test": "npm run cpsrc && npm run jest"
    },
    "jest": {
        "verbose": true,
        "testMatch": [
            "**/*.test.js"
        ],
        "transform": {}
    }
}
```

Use npm scripts to run tests

Running the tests is as easy as this now: `npm test` that copies the source modules to the test src directory and renaming the file suffix and run the test suite.



```
(node:877785) ExperimentalWarning: VM Modules is an experimental feature and might change at any time
(Use `node --trace-warnings ...` to show where the warning was created)
 PASS  ./htmlparser.test.js
  ✓ Language markers spread via several block elements. (4 ms)
  ✓ Only one closing language tag. (1 ms)
  ✓ Language tags in text and attribures, attributes without value. (1 ms)
  ✓ Already containing tiny tags for language markers.
  ✓ Html containing comments. (1 ms)
  ✓ Html containing a svg.
  ✓ Html with tex anotations.
  ✓ Html containing mathml elements.

Test Suites: 1 passed, 1 total
Tests:       8 passed, 8 total
Snapshots:   0 total
Time:        0.169 s, estimated 1 s
Ran all test suites.
```

Extend CI pipeline

In your `.github/workflows/moodle-plugin-ci.yml` add:

```
- name: JS Unit tests
  if: ${{ always() }}
  run: cd /home/runner/work/moodle-tiny_multilang2/moodle-tiny_multilang2/⍰
       moodle/lib/editor/tiny/plugins/multilang2/tests/js && ⍰
       npm install && npm test
```

This block might be inserted after the PHP Unit tests, before the Behat tests.

More fun with Mocha

Plugin: Tiny Cloze question editor

https://github.com/srobotta/moodle-tiny_cloze

JS Framework: Mocha (https://mochajs.org/)

Node.js must be installed!

Install and setup Mocha

Create a `package.json` with the following content:

```
{
    "type": "module",
    "devDependencies": {
        "mocha": "^11.0.1"
    },
    "scripts": {
        "cpsrc": "mkdir -p src && for i in ../../amd/src/*.js; ⍰
                  do t=${i##*/}; x=${t%*.js}; ⍰
                  cp $i src/${x}.mjs; done",
        "test": "npm run cpsrc && mocha --recursive ./*.js"
    }
}
```

and run `npm install`.

Write a Mocha test

Create a new file `test.js` with this content:

```
import * as assert from 'assert';
import * as cloze from './src/cloze.mjs';

describe('Test function getUuid()', function () {
  it('Check for length', function () {
    assert.equal(cloze.getUuid().length, 36);
  });
});
```

Run the test: `npm test`

Caveats

No Document Object Model (DOM) available, this is offered via APIs by the webbrowser.

To test DOM Manipulations use the Node package jsdom (https://github.com/jsdom/jsdom).

```
{
  "type": "module",
  "devDependencies": {
    "mocha": "^11.0.1",
    "jsdom": "^25.0.1"
  },
  "scripts": {
    "cpsrc": "mkdir -p src && for i in ../../amd/src/*.js ; do t=${i##*/}; x=${t%*.js}; cp $i src/${x}.mjs; done",
    "test": "npm run cpsrc && mocha --recursive ./*.js"
  }
}
```

Test a function with DOM

This is a function that uses a DOM Node element to check whether a class exists or not:

```
/**
 * Check if the node has the given class derived from the CSS object.
 *
 * @param {Node} node
 * @param {string} className
 * @returns {boolean}
 */
const hasClass = function(node, className) {
  return node.classList.contains(CSS[className]);
};
```

Test case using DOM

```
import * as assert from 'assert';
import * as jsdom from 'jsdom';
import * as cloze from './src/cloze.mjs';

describe('Test function hasClass()', function () {
  describe('create <i class="tiny_cloze_add">+</i>', function () {
    const dom = new jsdom.JSDOM(`<i class="tiny_cloze_add">+</i>`);
    const n = dom.window.document.querySelector('i');
    it('Check for existing class property ADD of CSS object.', function () {
      assert.equal(cloze.hasClass(n, 'ADD'), true);
    });
    it('Check for missing class property DELETE of CSS object.', function () {
      assert.equal(cloze.hasClass(n, 'DELETE'), false);
    });
  });
});
```

No Moodle available

JavaScript that uses the Moodle ecosystem cannot be used.

- No string loading
- No modals
- No TinyMCE
- No Moodle Settings

JS Modules to test might have to be rewritten, so that the code is separated from the code to test and the rest.

Thank you.