A Tool for Detecting and Refactoring the A?B*A Pattern in CSS

@leonardpunt
Leonard Punt
University of Amsterdam, The Netherlands
Q42, The Netherlands

@sjoerdvisscher
Sjoerd Visscher
Q42, The Netherlands

@grammarware
Vadim Zaytsev
University of Amsterdam, The Netherlands
Raincode, Belgium

The tool is available under the MIT license from this web location: http://leonardpunt.github.io/masterproject/tool.zip. This tool detects undoing style in CSS code and is able to apply refactoring opportunities to eliminate a subset of these instances of undoing style, while preserving the semantics of the web application.

Prerequisites:

- Firefox: https://www.mozilla.org/firefox/new/
- Firebug: https://addons.mozilla.org/nl/firefox/addon/firebug/
- npm: https://www.npmjs.com/
- The tool itself: https://leonardpunt.github.io/masterproject/

Setting it up:

- Unzip the downloaded tool
- Install dependencies by running `npm install`
- Build the tool by running `grunt`
- Two build artefacts can be found in the folder `dist`:
  - `tool.min.js`: the tool with minified source code (for deployment purposes)
  - `tool.js`: the tool with source code not minified (for debugging purposes)

Running it

- Open Firefox
- Browse to the webpage you would like to analyse
- Open Firebug
- Paste contents of `tool.min.js` in FB's Command Line
- Click the **Run** button
- Now these two entry points are available:
  - `detect()` — Detects undoing style in CSS. Results of the analysis are saved to a text file. After the analysis is completed, the user is prompted to download the results (`detectionsX.txt`).
  - `detectAndRefactor()` — Detects undoing style in CSS and applies refactoring opportunities. Results of the analysis are saved to a text file. Refactoring opportunities are applied dynamically. After the analysis and refactoring is completed, the user is prompted to download the results. The results comprise:
    - Detected undoing style (`detectionsX.txt`)
    - Detected semantic changes (`errors.txt`)
    - Refactored stylesheets (name of original stylesheet)

The tool can be used to analyse a single state of a web application, as well as multiple states (or pages) of a web application. In this section we will demonstrate both cases. Furthermore, we will explain the output format.

Example 1: Single state

- Open the web application in the desired state. For instance, https://leonardpunt.github.io/masterproject/ has only one state.
- Run the tool as described earlier.

Example 2: Multiple states

- Capture the multiple states of the web application (e.g. by opening it in a desired state and saving it).
- Create a new HTML document, using this template:

```
<!DOCTYPE html>
<html>
<head>
  <title>[site name]</title>
</head>
<body>
  <iframe src="[some-state.html]"
          width="90%"
          height="300px"></iframe>
  <iframe src="[another-state.html]"
          width="90%"
          height="300px"></iframe>
</body>
</html>
```

- Open this new HTML document.
- Run tool as described earlier, only now pass the value `true` to both entry points. I.e. `detect(true)` and/or `detectAndRefactor(true)`.

Example of output:

```
-- SMELL: 1 --
Smell: A-B(1)-A on margin-top
- initial: 0px (p)
- enclosed: 5px (#some-id)
- reset: 0px (#some-id.some-class)
- refactorable: true
- nodes: <p id="some-id" class="some-class">
```

The first line denotes the number of the undoing style pattern. The next line describes which pattern is found and on which CSS property, in this example we found the AB^1A pattern on the CSS property `margin-top`. The next lines show the CSS selector which sets the style initially, the CSS selector(s) that are enclosed and the CSS selector that resets the style. The last but one line shows if the pattern can be refactored. The last line shows the nodes that applied to the detected pattern.

Quality. All code conforms to the JSHint rules. Unit tests are included as well. Verify this by running `grunt test`.