Loud

Release 0.9.2

Aug 12, 2022

Contents

Web accessibility testing helper

CHAPTER 1

What is it?

Loud is a JavaScript library for browser, which helps track regression of accessibility.

Loud ships under terms of the MIT License.

CHAPTER 2

Why do I need it?

You break HTML pages on elements. Each element you can create in different ways. For example, you can create a button like this (with a little bit of JavaScript):

```
<i role="button" aria-label="Join"></i>
```

From accessibility point of view, this is a button. Later, you decide to change the button to something like this:

```
<button>Join</button>
```

From accessibility point of view, this is also a button and both buttons are the same.

Loud knows how elements look like from the accessibility point of view. You can use this information to track accessibility regression of your web pages.

CHAPTER 3

Table of Contents

3.1 Getting Started

Get a release tarball, or clone the repository, or use npm and browserify, or bower:

```
bower install loud --save-dev
```

Add `./dist/loud.js` to a testing page:

```
<script src="/path/to/loud/dist/loud.js"></script>
```

Note: You should use es5-shim in old browsers.

Test with Loud (using Jasmine, for example):

```
describe('loud', function() {
    beforeEach(function() {
        this.button = document.createElement('button');
        this.button.innerHTML = 'Join';
        document.body.appendChild(this.button);
    });

    beforeEach(function() {
        document.body.removeChild(this.button);
    });

    it('works', function() {
        expect(loud.say(this.button)).toEqual([
            'Join', 'button'
        ]);
    });
});
```

Changed in version 0.9.0: Adding elements to DOM is mandatory. Otherwise, an element visibility will not be properly detected.

3.2 API

loud

Type Object

loud.**VERSION**

Type String

loud.**FORCE_VALID_MARKUP**

Type Boolean

Default true

Since 0.9.0

Force markup to be valid. Set to `false`, to handle invalid markup as valid.

loud.**say**(*node*)

Arguments

- **node** (*Object|Object[]*) – DOM element or array of DOM elements

Return type Array of Strings

Returns Words

Transform a DOM element to words.

loud.**warn**(*message*)

Arguments

- **message** (*String*) – Error message

Since 0.9.0

Warn about failed validation.

loud.**error**(*message*)

Arguments

- **message** (*String*) – Error message

Throws loud.ValidationError

Since 0.9.0

Throw validation error.

loud.**ValidationError**(*message*)

Arguments

- **message** (*String*) – Error message

Type Function

Since 0.9.0

Validation error.

3.3 Examples

3.3.1 Document Structure

```
<body>
  <header>
    <img src="/logo.png" alt="My Company"/>

    <form>
      <input type="search" aria-label="Query">
      <button>Search</button>
    </form>
  </header>

  <main>
    <h1>Main Theme</h1>

    <p>Main Content.</p>
  </main>

  <footer>
    Copyright
  </footer>
</body>
```

```
describe('everything', function() {
    it('works', function() {
        var page = document.getElementsByTagName('body')[0];

        expect(loud.say(page)).toEqual([
            'banner',
            'My Company', 'img',
            'Query', 'textbox',
            'Search', 'button',
            'banner end',

            'main',
            'Main Theme', 'heading', 'level', '1',
            'Main Content.',
            'main end',

            'contentinfo',
            'Copyright',
            'contentinfo end'
        ]);
    });
});
```

3.3.2 Controls

```
<input type="checkbox" id="c1"/>
```

```
describe('checkbox', function() {
    beforeEach(function() {
```

(continues on next page)

(continued from previous page)

```
        this.checkbox = document.createElement('input');
        this.checkbox.type = 'checkbox';
        document.body.appendChild(this.checkbox);
    });

    afterEach(function() {
        document.body.removeChild(this.checkbox);
    });

    it('not checked', function() {
        expect(loud.say(this.checkbox)).toEqual([
            'checkbox', 'not checked'
        ]);
    });

    it('checked', function() {
        this.checkbox.click();

        expect(loud.say(this.checkbox)).toEqual([
            'checkbox', 'checked'
        ]);
    });
});
```

3.4 Predicting Output

3.4.1 Predicting Accessible Name

3.4.2 Predicting Roles

Widgets

Widgets does not have content.

Role	Output	HTML
alertdialog	`['alertdialog']`	
button	`['button']`	`<button>`, `<input type=button>`
checkbox	`['checkbox']`	`<input type=checkbox>`
dialog	`['dialog']`	
gridcell	`['gridcell']`	`<td>`
link	`['link']`	`<a>`
menuitem	`['menuitem']`	
menuitemcheckbox	`['menuitemcheckbox']`	
menuitemradio	`['menuitemradio']`	
option	`['option']`	`<option>`
progressbar	`['progressbar']`	`<progress>`
radio	`['radio']`	`<input type=radio>`
scrollbar	`['scrollbar']`	
slider	`['slider']`	`<input type=range>`
spinbutton	`['spinbutton']`	
tab	`['tab']`	
tabpanel	`['tabpanel']`	
textbox	`['textbox']`	`<input type=text>`
tooltip	`['tooltip']`	
treeitem	`['treeitem']`	

Composite widgets

Composite widgets have content.

Tables are special.

Role	Output	HTML
combobox	`['combobox', ...]`	
grid	`['grid', ...]`	
listbox	`['listbox', ...]`	`<select>`
menu	`['menu', ...]`	
menubar	`['menubar', ...]`	
radiogroup	`['radiogroup', ...]`	
tablist	`['tablist', ...]`	
tree	`['tree', ...]`	
treegrid	`['treegrid', ... 'treegrid end']`	

Document Structure

Role	Output	HTML
article	`['article', ... 'article end']`	`<article>`
column-header	`['columnheader', ..., 'columnheader end']`	`<th>`
definition	`['difinition', ..., 'definition end']`	
directory	`['directory', ... 'directory end']`	
document	`['document', ..., 'document end']`	`<body>`
group	`['group', ..., 'group end']`	
heading	`['heading']`	`<h1>--<h6>`
img	`['img']`	`<img>`
list	`['list', ..., 'list end']`	`<ul>, <ol>`
listitem	`['listitem']`	`<li>`
math	`['math']`	
note	`['note', ..., 'note end']`	
region	`['region', ..., 'region end']`	
row	`['row', ...]`	`<tr>`
rowgroup	`['rowgroup', ...]`	`<thead>, <tfoot>, <tbody>`
rowheader	`['rowheader', ...]`	`<th scope=row>`
separator	`['separator']`	`<hr>`
toolbar	`['toolbar', ..., 'toolbar end']`	

Landmarks

Role	Output	HTML
application	`['application', ..., 'application end']`	
banner	`['banner', ..., 'banner end']`	`<header>`
complementary	`['complementary', ..., 'complementary end']`	`<aside>`
contentinfo	`['contentinfo', ..., 'contentinfo end']`	`<footer>`
form	`['form', ..., 'form end']`	
main	`['main', ..., 'main end']`	`<main>`
navigation	`['navigation', ..., 'navigation end']`	`<nav>`
search	`['search', ..., 'search end']`	

Live regions

Role	Output
alert	`['alert', ..., 'alert end']`
log	`['log', ..., 'log end']`
marquee	`['marque', ..., 'marque end']`
status	`['status', ..., 'status end']`
timer	`['timer', ..., 'timer end']`

Abstract Roles

No output for abstract roles.

Role	Output
command	`[...]`
composite	`[...]`
input	`[...]`
landmark	`[...]`
range	`[...]`
roletype	`[...]`
section	`[...]`
sectionhead	`[...]`
select	`[...]`
structure	`[...]`
widget	`[...]`
window	`[...]`

Presentation

Role	Output
presentation	`[...]`

3.4.3 Predicting States and Properties

Widget States

State	HTML	Positive	Negative
aria-checked	checked	`[role, 'checked']`	`[role, 'not checked']`
aria-disabled	disabled	`[role, 'disabled']`	
aria-expanded		`[role, 'expanded']`	`[role, 'collapsed']`
aria-hidden	hidden	`[role, 'hidden']`	
aria-invalid		`[role, 'invalid']`	
aria-pressed		`[role, 'pressed']`	`[role, 'not pressed']`
aria-selected	selected	`[role, 'selected']`	

Widget Properties

Property	Output
aria-autocomplete	`[role, 'autocomplete', value]`
aria-haspopup	`[role, 'haspopup']`
aria-label	See accessible name
aria-level	`[role, 'level', value]`
aria-multiline	`[role, 'multiline']`
aria-multiselectable	`[role, 'multiselectable']`
aria-orientation	`[role, 'orientation', value]`
aria-readonly	`[role, 'readonly']`
aria-required	`[role, 'required']`
aria-sort	`[role, 'sort order', value]`
aria-valuemax	See ranges
aria-valuemin	See ranges
aria-valuenow	See ranges
aria-valuetext	See ranges

Live Region States

State	Positive	Negative
aria-busy	`[role, 'busy']`	

Live Region Properties

Property	Output
aria-atomic	`[role, 'atomic']`
aria-live	`[role, 'live', value]`
aria-relevant	`[role, 'relevant', value]`

Drag-and-Drop States

State	Output
aria-grabbed	`[role, 'grabbed'] [role, 'grabbable']`

Drag-and-Drop Properties

Property	Output
aria-dropeffect	`[role, 'dropeffect', value]`

Relationship Properties

Property	Output
aria-activedescendant	See active descendant
aria-controls	`[role, 'controls', value]`
aria-describedby	Accessible name of the linked elements
aria-flowto	`[role, 'flowto', value]`
aria-labelledby	See accessible name
aria-owns	`[role, 'owns', value]`
aria-posinset	`[role, value, 'of', setsize]`
aria-setsize	`[role, 'of', setsize, 'items']`

Order of States and Properties

1. multiline
2. orientation
3. posinset
4. setsize
5. invalid
6. disabled
7. level
8. sort order
9. checked
10. not checked
11. expanded
12. collapsed
13. pressed
14. not pressed
15. selected
16. grabbed
17. grabbable
18. busy
19. required
20. readonly
21. multiselectable
22. haspopup
23. autocomplete
24. dropeffect
25. live

26. relevant
27. atomic
28. controls
29. owns
30. flowto
31. described by
32. active descendant

The order is public API.

3.5 Developing Loud

Clone the repository and install dependencies:

```
git clone https://github.com/ruslansagitov/loud.git
npm install
```

Linters

Install linters globally:

```
npm install --global eslint
```

Now you can run linters:

```
npm run-script lint
```

Note: If linting fails, try to install specific versions of linters. You can find version numbers in the Continues Integration configuration.

Checking documentation

Install tools:

```
npm install --global write-good
```

And check writing:

```
npm run-script check-writing
```

Testing in browsers

Install karma plugins for testing in specific browsers:

```
npm install karma-firefox-launcher
npm install karma-phantomjs-launcher
export KARMA_BROWSERS=Firefox,PhantomJS
```

Note: Karma launchers are not installing automatically. You should choose yourself in which browsers you will test.

After installing launchers, build the sources:

```
mkdir build
npm run-script build
```

Now you can test with karma:

```
npm run-script karma-test
```

Running all the tests

```
npm test
```

Warning: You should have already installed all the dependencies, linters and karma launchers at this point!

Getting code coverage

```
npm run-script coverage
```

The coverage is in the *coverage* directory.

Warning: Code coverage should not drop below 100%.

Generating JSDoc

```
npm install --global jsdoc
npm run-script jsdoc
```

The JavaScript documentation is in the *jsdoc* directory.

Generating documentation

You need to install sphinx (a python-tool) to generate documentation:

```
pip install sphinx
```

Now you can generate documentation:

```
npm run-script doc
```

Deploying

Just before release, generate distibution files:

```
npm run-script build
```

Important: Don't forget to commit the generated files for bower.

Releasing a new version

1. Find all the strings with the previous version and change it for the new version
2. Commit changes with the commit message: "Bump version"
3. Tag commit as 'vX.Y.Z'
4. Push changes with tags

Index

L