ng add @angular-eslint/schematics

ℹ Using package manager: npm

✔ Found compatible package version: @angular-eslint/schematics@14.2.5.

✔ Package information loaded.

*npm install prettier — save-dev*

npm install prettier eslint-plugin-prettier eslint-config-prettier --save-dev

The package @angular-eslint/schematics@14.2.5 will be installed and executed.

Would you like to proceed? Yes

✔ Packages successfully installed.

All @angular-eslint dependencies have been successfully installed 🎉

Please see https://github.com/angular-eslint/angular-eslint for how to add ESLint configuration to your project.

We detected that you have a single project in your workspace and no existing linter wired up, so we are configuring ESLint for you automatically.

Please see https://github.com/angular-eslint/angular-eslint for more information.

CREATE .eslintrc.json (984 bytes)

UPDATE package.json (1511 bytes)

UPDATE angular.json (3447 bytes)

✔ Packages installed successfully.

Next, we will run the command below to validate the ESLint installation and configuration:

npm run lint

> angular-eslint@1.0.0 lint /home/rodrigokamada/angular-eslint

> ng lint

Linting "angular-eslint"...

All files pass linting.

*ng lint — fix*

npm run format && npm run lint

How to diable:

n VSCode, go to

File >> preferences >> settings

Type **'eslint quiet'** in the search bar and click the check box for quiet mode.

In quiet mode, eslint would ignore basic errors.

**Angular ESLint Rules for Accessible HTML Content**

[**Angular ESLint Rules for Accessibility (3 Part Series)**](https://dev.to/sandikbarr/series/20450)

[1Angular ESLint Rules for Keyboard Accessibility](https://dev.to/angular/angular-eslint-rules-for-keyboard-accessibility-236f)

[2Angular ESLint Rules for ARIA](https://dev.to/angular/angular-eslint-rules-for-aria-3ba1)

[3Angular ESLint Rules for Accessible HTML Content](https://dev.to/angular/angular-eslint-rules-for-accessible-html-content-kf5)

Content accessibility for built-in HTML elements is the third and final category in this series on Angular ESLint accessibility rules. These rules validate several HTML attributes that developers commonly overlook regarding accessibility. They check for alt text on images, accessible content in buttons, links, and headings, and proper form labels and table header associations. Angular ESLint accessibility rules provide immediate guidance on accessibility best practices right in the code, resulting in more accessible and user-friendly Angular applications.

**The Rules**

Previous articles in this series discussed how to add Angular ESLint and configure these rules in .eslintrc.json config files, so let's get straight into the rules:

* [Alt Text](https://dev.to/angular/angular-eslint-rules-for-accessible-html-content-kf5#rule-alt-text)

@angular-eslint/template/accessibility-alt-text

* [Elements Have Content](https://dev.to/angular/angular-eslint-rules-for-accessible-html-content-kf5#rule-elements-have-content)

@angular-eslint/template/accessibility-elements-content

* [Label Has Associated Control](https://dev.to/angular/angular-eslint-rules-for-accessible-html-content-kf5#rule-label-has-associated-control)

@angular-eslint/template/accessibility-label-has-associated-control

* [Table Scope](https://dev.to/angular/angular-eslint-rules-for-accessible-html-content-kf5#rule-table-scope)

@angular-eslint/template/accessibility-table-scope

* [No Distracting Elements](https://dev.to/angular/angular-eslint-rules-for-accessible-html-content-kf5#rule-no-distracting-elements)

@angular-eslint/template/no-distracting-elements

* [Button Has Type](https://dev.to/angular/angular-eslint-rules-for-accessible-html-content-kf5#bonus-rule-button-has-type)

@angular-eslint/template/button-has-type

**Rule: Alt Text**

[**@angular-eslint/template/accessibility-alt-text**](https://github.com/angular-eslint/angular-eslint/blob/main/packages/eslint-plugin-template/docs/rules/accessibility-alt-text.md)

The accessibility-alt-text rule validates that all images have alternative text. Images must have an alt attribute to meet [WCAG 2.1 Success Criterion 1.1.1: Non-text Content](https://www.w3.org/WAI/WCAG21/Understanding/non-text-content.html) that states all non-text content should have a text alternative that serves the equivalent purpose.

Follow these guidelines for providing meaningful and concise alt text:

* Alt text should express the relevant detail in an image to stand in for the same purpose, meaning, and intent.
* Alt text should not be redundant or repeat information from the image caption.
* Alt text should be short and to the point, ideally 150 characters or less.

When there is no alt attribute on an <img>, some screen readers will announce the image src instead. Including an empty alt attribute for decorative images indicates to screen readers that these images do not convey additional meaning or information:

<img alt="" src="decorative.gif">

In addition to checking for the alt attribute on <img> elements, the accessibility-alt-text rule also validates <input type="image">, <area>, and <object> elements, which support these attributes as alternative text:

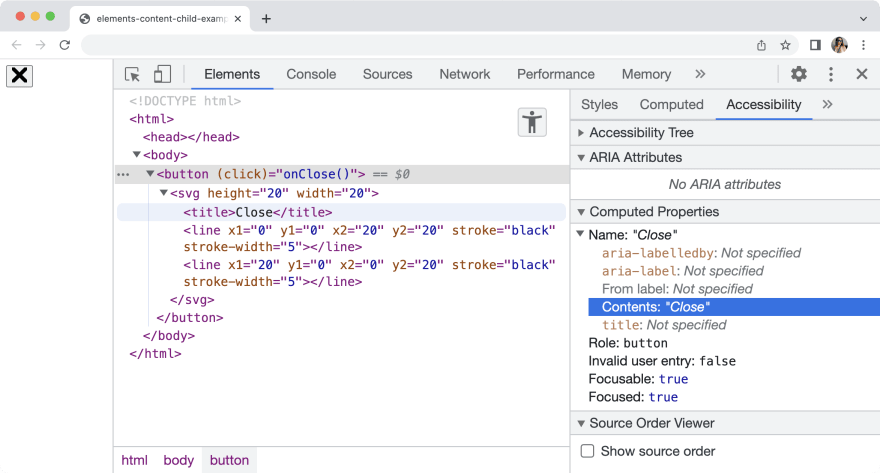
| **HTML element** | **alternative text attributes** |
| --- | --- |
| [<img>](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/img) | alt |
| [<input type="image">](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/input/image) | alt, aria-label, aria-labelledby |
| [<area>](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/area) | alt, aria-label, aria-labelledby |
| [<object>](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/object) | aria-label, aria-labelledby, title |

**Rule: Elements Have Content**

[**@angular-eslint/template/accessibility-elements-content**](https://github.com/angular-eslint/angular-eslint/blob/main/packages/eslint-plugin-template/docs/rules/accessibility-elements-content.md)

The accessibility-elements-content rule ensures that <a>, <button>, and <h1> - <h6> elements have content. Screen readers announce these elements by their role and name (and level for headings.) Links, buttons, and headings are semantic elements with inherent meaning and implicit roles, and the [accessible name](https://www.w3.org/WAI/ARIA/apg/practices/names-and-descriptions/) for these elements comes from their content, title, or aria-label.

A button, link, or heading name may also come from a child element with plain text content, a title, or aria-label. Don't add redundant and unnecessary aria-labels where the accessible name comes from a child element. Use the [Accessibility pane in Chrome DevTools](https://developer.chrome.com/docs/devtools/accessibility/reference/#pane) to inspect an element's computed name:

[](https://res.cloudinary.com/practicaldev/image/fetch/s--mfKIU_eh--/c_limit%2Cf_auto%2Cfl_progressive%2Cq_auto%2Cw_880/https:/dev-to-uploads.s3.amazonaws.com/uploads/articles/ze46w6rhfu4ze6otr0o3.png)

**Rule: Label Has Associated Control**

[**@angular-eslint/template/accessibility-label-has-associated-control**](https://github.com/angular-eslint/angular-eslint/blob/main/packages/eslint-plugin-template/docs/rules/accessibility-label-has-associated-control.md)

The HTML [<label>](https://www.w3schools.com/tags/tag_label.asp) element gives an accessible name to form controls. Labels are important for helping users understand the purpose of <input>, <select>, <textarea>, <output>, <meter>, and <progress> elements.

* A control can have more than one <label>.
* A <label> cannot be associated with more than one control.
* Do not use <label> independently without an associated control.

A <label> is associated with a form control with either an implicit or explicit association, and both styles are widely supported. Nesting a control inside <label> creates an **implicit** association between the label and control:

<label>

<input type="checkbox">

Blue

</label>

Whereas, assigning an id to the control and using a matching for attribute on the <label> creates an **explicit** association:

<label for="city">City</label>

<input id="city" type="text">

The accessibility-label-has-associated-control rule determines if a label has an implicit association with a control nested inside the label or if the label has the for attribute. The rule does not look elsewhere in the template for a control with an id matching the label's for attribute and does not validate explicit label and control pairings. This is a natural limitation of static code analysis tools like Angular ESLint, so you’ll want to use additional testing techniques to validate the compiled application. The for attribute could be a bound value, and Angular ESLint template rules do not validate bound values. Rules are generally limited to validating individual template nodes and their children, and the control could even be in another component's template separate from the label.

**Configuration Options for Label Has Associated Control**

The accessibility-label-has-associated-control rule supports the [configuration options](https://github.com/angular-eslint/angular-eslint/blob/main/packages/eslint-plugin-template/docs/rules/accessibility-label-has-associated-control.md#rule-options) labelComponents and controlComponents. The labelComponents option adds validation for custom label components. The controlComponents option configures the rule to recognize an implicit association on custom controls nested inside a label.

**Option: Custom Label Components**

Configure "labelComponents" for "my-custom-label" with "forControl":

"@angular-eslint/template/accessibility-label-has-associated-control": [

"error",

{

"labelComponents": [

{

"selector": "my-custom-label",

"inputs": ["forControl"]

}

]

}

]

Validation on a custom label component to either require the forControl input or to have a nested control:

<my-custom-label forControl="myId">

Custom

</my-custom-label>

<input id="myId" type="text"/>

<my-custom-label>

Custom

<input type="text"/>

</my-custom-label>

**Option: Custom Control Components**

Configure "controlComponents" for "my-custom-control":

"@angular-eslint/template/accessibility-label-has-associated-control": [

"error",

{

"controlComponents": ["my-custom-control"]

}

]

Validation on a custom control nested inside a label with an implicit association:

<label>

Custom

<my-custom-control></my-custom-control>

</label>

**Rule: Table Scope**

[**@angular-eslint/template/accessibility-table-scope**](https://github.com/angular-eslint/angular-eslint/blob/main/packages/eslint-plugin-template/docs/rules/accessibility-table-scope.md)

The accessibility-table-scope rule validates the scope attribute for table headers. The scope attribute specifies which cells belong with a table header (<th>.) Table header scope accepts the following values:

* **row**: associates a table header with all the cells in that row.
* **col**: associates a table header with all the cells in that column.
* **rowgroup**: associates a table header that spans multiple rows with all the cells in that row group.
* **colgroup**: associates a table header that spans multiple columns with all the cells in that column group.

When table header scope is not specified, browsers and assistive technologies infer the relationship between table headers and their cells. Tables with headers in a single row or column do not need the scope attribute. [More complex tables](https://www.w3.org/WAI/tutorials/tables/) with irregular, multi-level, or both row and column headers should explicitly define which cells belong to which headers by using the scope attribute on their table headers or the id attribute on table headers with the headers attribute on table cells.

**Rule: No Distracting Elements**

[**@angular-eslint/template/no-distracting-elements**](https://github.com/angular-eslint/angular-eslint/blob/main/packages/eslint-plugin-template/docs/rules/no-distracting-elements.md)

The no-distracting-elements rule disallows usage of the <blink> and <marquee> elements, which are both deprecated and no longer recommended for use.

The <blink> element is more than distracting. Flashing content can trigger seizures in people with photosensitive epilepsy. [WCAG 2.1 Success Criterion 2.3.1](https://www.w3.org/WAI/WCAG21/Understanding/three-flashes-or-below-threshold.html) dictates that there are no more than three flashes in a one second period or that the flash is below the threshold.

The scrolling content in a <marquee> element can create barriers for anyone who struggles with moving objects or people with cognitive disabilities like attention deficit disorder. [WCAG 2.1 Success Criterion 2.2.2](https://www.w3.org/WAI/WCAG21/Understanding/pause-stop-hide.html) states that users must be able to stop or hide any moving, blinking, scrolling, or auto-updating information.

**Bonus Rule: Button Has Type**

[**@angular-eslint/template/button-has-type**](https://github.com/angular-eslint/angular-eslint/blob/main/packages/eslint-plugin-template/docs/rules/button-has-type.md)

I mention this last rule as a bonus because it is not specifically an accessibility rule but is commonly missed and can lead to surprising functionality. The button-has-type rule checks for the type attribute on HTML <button> elements. A <button> inside a <form> without a type acts as a submit button and submits the form when pressed:

<form>

<label><input type="checkbox">Yes</label>

<button>Submits the Form</button>

<button type="button">Not a Submit</button>

<button type="reset">Resets the Form</button>

<button type="submit">Submits the Form</button>

</form>

Formatters:

ESLint comes with several built-in formatters to control the appearance of the linting results, and supports third-party formatters as well.

You can specify a formatter using the --format or -f flag in the CLI. For example, --format json uses the json formatter.

The built-in formatter options are:

* [checkstyle](https://eslint.org/docs/latest/use/formatters/#checkstyle)
* [compact](https://eslint.org/docs/latest/use/formatters/#compact)
* [html](https://eslint.org/docs/latest/use/formatters/#html)
* [jslint-xml](https://eslint.org/docs/latest/use/formatters/#jslint-xml)
* [json-with-metadata](https://eslint.org/docs/latest/use/formatters/#json-with-metadata)
* [json](https://eslint.org/docs/latest/use/formatters/#json)
* [junit](https://eslint.org/docs/latest/use/formatters/#junit)
* [stylish](https://eslint.org/docs/latest/use/formatters/#stylish)
* [tap](https://eslint.org/docs/latest/use/formatters/#tap)
* [unix](https://eslint.org/docs/latest/use/formatters/#unix)
* [visualstudio](https://eslint.org/docs/latest/use/formatters/#visualstudio)

Example Source

Examples of each formatter were created from linting fullOfProblems.js using the .eslintrc.json configuration shown below.

fullOfProblems.js:

function addOne(i) {

if (i != NaN) {

return i ++

} else {

return

}

};

.eslintrc.json:

{

"extends": "eslint:recommended",

"rules": {

"consistent-return": 2,

"indent" : [1, 4],

"no-else-return" : 1,

"semi" : [1, "always"],

"space-unary-ops" : 2

}

}

Tests the formatters with the CLI:

npx eslint --format <Add formatter here> fullOfProblems.js

EsLint Integrations:

1. [Editors](https://eslint.org/docs/latest/use/integrations" \l "editors)
2. [Build tools](https://eslint.org/docs/latest/use/integrations#build-tools)
3. [Command Line Tools](https://eslint.org/docs/latest/use/integrations#command-line-tools)
4. [Source Control](https://eslint.org/docs/latest/use/integrations#source-control)
5. [Testing](https://eslint.org/docs/latest/use/integrations#testing)
6. [Other Integration Lists](https://eslint.org/docs/latest/use/integrations#other-integration-lists)

Editors

* Sublime Text 3:
  + [SublimeLinter-eslint](https://github.com/roadhump/SublimeLinter-eslint)
  + [Build Next](https://github.com/albertosantini/sublimetext-buildnext)
* Vim:
  + [ALE](https://github.com/w0rp/ale)
  + [Syntastic](https://github.com/vim-syntastic/syntastic/tree/master/syntax_checkers/javascript)
* Emacs: [Flycheck](http://www.flycheck.org/) supports ESLint with the [javascript-eslint](http://www.flycheck.org/en/latest/languages.html" \l "javascript) checker.
* Eclipse Orion: ESLint is the [default linter](https://dev.eclipse.org/mhonarc/lists/orion-dev/msg02718.html)
* Eclipse IDE: [Tern ESLint linter](https://github.com/angelozerr/tern.java/wiki/Tern-Linter-ESLint)
* TextMate 2:
  + [eslint.tmbundle](https://github.com/ryanfitzer/eslint.tmbundle)
  + [javascript-eslint.tmbundle](https://github.com/natesilva/javascript-eslint.tmbundle)
* Atom:
  + [linter-eslint](https://atom.io/packages/linter-eslint)
  + [fast-eslint-8](https://atom.io/packages/fast-eslint-8)
* IntelliJ IDEA, WebStorm, PhpStorm, PyCharm, RubyMine, and other JetBrains IDEs: [How to use ESLint](https://www.jetbrains.com/help/webstorm/eslint.html)
* Visual Studio: [Linting JavaScript in VS](https://learn.microsoft.com/en-us/visualstudio/javascript/linting-javascript?view=vs-2022)
* Visual Studio Code: [ESLint Extension](https://marketplace.visualstudio.com/items?itemName=dbaeumer.vscode-eslint)
* Brackets: Included and [Brackets ESLint](https://github.com/brackets-userland/brackets-eslint)

Build tools

* Grunt: [grunt-eslint](https://www.npmjs.com/package/grunt-eslint)
* Gulp: [gulp-eslint](https://www.npmjs.com/package/gulp-eslint)
* Mimosa: [mimosa-eslint](https://www.npmjs.com/package/mimosa-eslint)
* Broccoli: [broccoli-eslint](https://www.npmjs.com/package/broccoli-eslint)
* Browserify: [eslintify](https://www.npmjs.com/package/eslintify)
* Webpack: [eslint-webpack-plugin](https://www.npmjs.com/package/eslint-webpack-plugin)
* Rollup: [@rollup/plugin-eslint](https://www.npmjs.com/package/@rollup/plugin-eslint)
* Ember-cli: [ember-cli-eslint](https://www.npmjs.com/package/ember-cli-eslint)
* Sails.js: [sails-hook-lint](https://www.npmjs.com/package/sails-hook-lint), [sails-eslint](https://www.npmjs.com/package/sails-eslint)
* Start: [@start/plugin-lib-eslint](https://www.npmjs.com/package/@start/plugin-lib-eslint)
* Brunch: [eslint-brunch](https://www.npmjs.com/package/eslint-brunch)

Command Line Tools

* [ESLint Watch](https://www.npmjs.com/package/eslint-watch)
* [Code Climate CLI](https://github.com/codeclimate/codeclimate)
* [ESLint Nibble](https://github.com/IanVS/eslint-nibble)

Source Control

* [Git Precommit Hook](https://coderwall.com/p/zq8jlq/eslint-pre-commit-hook)
* [Git pre-commit hook that only lints staged changes](https://gist.github.com/dahjelle/8ddedf0aebd488208a9a7c829f19b9e8)
* [overcommit Git hook manager](https://github.com/brigade/overcommit)
* [Mega-Linter](https://nvuillam.github.io/mega-linter): Linters aggregator for CI, [embedding eslint](https://nvuillam.github.io/mega-linter/descriptors/javascript_eslint/)

Testing

* Mocha.js: [mocha-eslint](https://www.npmjs.com/package/mocha-eslint)