JavaScript

JavaScript is supported out of the box in Emacs, see `js-mode`.

See also TypeScript, Vuejs.

Nicolas Petton, the author of Indium and core contributor to Emacs, has a blog post series on using Emacs for Javascript development.

javascript modes

js-mode

js-mode is the default javascript mode for emacs. See `js-mode`.

js2-mode

`js2-mode` (in GNU ELPA) by Steve Yegge is one of the most complete JavaScript modes. It boasts **run-time validation** of JavaScript.

As of 20150909, it derives from js-mode. That means the former function will run js-mode-hook, as well as js2-mode-hook. The key bindings will default to js-mode-map where they're not set in js2-mode-map. And in Emacs 25 or later (including the snapshot builds), js2-mode uses the indentation code from js-mode.

See the latest visible changes.

jsx support

js2-mode supports jsx with `js2-jsx-mode`. It supports indentation of JSXElement expressions wrapped within parentheses or as function arguments. Indentation is customizable via `sgml-attribute-offset`.

js3-mode

js3-mode is a fork of js-mode and js2-mode that supports comma-first style and other quirks.

The goal of this project was to get a javascript mode working that supports npm style, but it turns out this mode is compatible with other styles as well.

Notably, js3-mode does not support js2-mode's bounce-indent, though it does support several popular indentation styles.

Tern, the "intelligent javascript tooling"

Tern is a stand-alone code-analysis engine for JavaScript. It is intended to be used with a code editor plugin, such as Emacs, to enhance the editor's support for intelligent JavaScript editing. Features provided are:

- Autocompletion on variables and properties
- Function argument hints
- Querying the type of an expression
- Finding the definition of something
- Automatic refactoring

To see what Tern is all about, you should try the online demo !

Tern is open-source (MIT license), written in JavaScript, and capable of running both on node.js and in the browser.

See the installation instructions for Emacs' Tern mode .

Indium: a REPL, inspector, stepping debugger and more

Indium, in MELPA, connects to a browser tab or nodejs process and provides

many features for JavaScript development, including:

- a REPL (with auto completion) & object inspection;
- an inspector, with history and navigation;
- a scratch buffer (**M-x indium-scratch**);
- JavaScript evaluation in JS buffers with **indium-interaction-mode**;
- a stepping Debugger, similar to edebug, or cider.

See its installation instructions.

It works with a Chrome(ium) and Nodejs backend. The firefox backend is in the TODO list.

Javascript linters

- flymake-jslint
- eslintd-fix - for on the fly checking with eslint_d, a faster eslint.

Typescript

For Typescript support, see TypeScript.

React JS

rjsx-mode is a special mode for editing JSX files. We get js2-mode features plus proper syntax checking and highlighting of JSX code blocks.

AngularJS integration

imenu integration

See https://github.com/redguardtoo/emacs.d/blob/master/lisp/init-javascript.el

Tern integration

See http://ternjs.net/doc/manual.html#plugin_angular

and a company backend: https://github.com/proofit404/company-tern/

Highlighting Angular directives in templates

See https://github.com/omouse/angularjs-mode/blob/master/angular-html-mode.el

JavaScript REPL

Have a look at mozrepl - a REPL for interacting with an external web browser's internal JavaScript engine. However MozRepl seems to be unmaintained. There is swank-js which is browser independent and based on Node.JS and SLIME.

Mix html and Javascript

See the packages that allow to have multiple modes in the same buffer at the same time:

- http://web-mode.org/
- https://github.com/vspinu/polymode
- https://github.com/purcell/mmm-mode
- https://github.com/fgallina/multi-web-mode

See also

The following packages are available on MELPA.

Linters

You can use flymake-jslint or flymake-gjslint to lint your javascript code.

Run javascript in an inferior process window

js-comint.el let's you run an inferior javascript process in emacs, and defines a few functions for sending javascript input to it quickly.

Live browser eval of JavaScript and html, possibly every time a buffer changes

This can be accomplished with skewer-mode or livid-mode.

Beautify HTML, CSS and JavaScript/JSON

This will be done with web-beautify.