Js2Mode

Overview

js2-mode by SteveYegge is arguably the best JavaScript mode available for emacs. It has very accurate syntax highlighting, supports newer JavaScript extensions implemented in SpiderMonkey, and highlights syntax errors as you type.

Obtaining and installing

js2-mode does not ship with Emacs, but it's included in GNU ELPA.

Development of js2-mode has moved to mooz's repository. Steve's original repository hasn't seen a lot of updates for quite some time. Stable snapshots get committed to the elpa branch.

The latest version requires Emacs 24, and has had some non-essential features removed, like auto-insertion of parentheses, since you are better off getting those from either Emacs 24's built-in electric-pair-mode, or from autopair.el.

Smart Tabs

Smart Tabs: Js2Mode

Js2Mode and jslint

The js2-mode has a nice feature in that it highlights undeclared variables, but sometimes you want to ignore these for certain symbols (perhaps you know the global is defined properly in another file). The jslint tool will look for a globals comment at the top of a file and consider any symbols defined in there

to be valid globals, the following elisp code installs a hook into js2-mode so it that it respects the same declaration – TimMeadowcroft

JSLint-style globals declaration is supported natively since 20130510. So using the code below is only advised for people who stay with an older version, e.g. for Emacs 23 compatibilty. – DmitryGutov

```
(add-hook 'js2-post-parse-callbacks
          (lambda ()
            (when (> (buffer-size) 0)
              (let ((btext (replace-regexp-in-string
                            ": *true" " "
                            (replace-regexp-in-string "[\n\t ]+" " " (buff
                (mapc (apply-partially 'add-to-list 'js2-additional-externs
                      (split-string
                       (if (string-match "/\\* *global *\\(.*?\\) *\\*/" b
                       " *, *" t))
                ))))
```

The jslint tool has an advanced version of this feature - the globals list can specify symbols that are valid globals (“name: true”) but it can also declare that some symbols are valid as long as you don’t assign to them (“name: false”), and it also seems that the globals comments can be placed inside a function and there the checking will respect the function scope. The js2-mode doesn’t seem (to me) to have either of these concepts inherently built in, so the version below will consider all global decls to have file scope, and it will respect “name: true” symbols but it will ignore “name: false” symbols (i.e. they’ll still be reported as invalid globals when used) - you could change this by adjusting the regexps below, but extending js2-mode to understand these more advanced concepts for looking at undeclared symnbols is probably a

more major task – TimMeadowcroft

Some comment on the use of the function `buffer-substring-no-properties` in the example code given above in case someone wants to see how the code works. The function `buffer-substring` and `buffer-substring-no-properties` have some gotchas. The expression `(buffer-substring-no-properties 1 (buffer-size))` captures the whole text except the last character, and causes args-out-of-range error in an empty buffer. The expression `(buffer-substring-no-properties 1 (1+ (buffer-size)))` captures the whole text but causes args-out-of-range error in a narrowed buffer. Finally, the expression `(save-restriction (widen) (buffer-substring-no-properties (point-min) (point-max)))` captures the whole text fine even in a narrowed buffer. – JisangYoo

js2-mode + google closure

If you want js2-mode to use names from goog.provide/goog.require you can add this to your .emacs:

```
(add-hook 'js2-post-parse-callbacks
          (lambda ()
            (let ((buf (buffer-string))
                  (index 0))
              (while (string-match "\\(goog\\.require\\|goog\\.provide\\)('\\(
                (setq index (+ 1 (match-end 0)))
                (add-to-list 'js2-additional-externs (match-string 2 buf))))))
```

Of course, it is only useful if you keep direct relationship between strings in goog.provide/goog.require and actual objects. – AdamRzepka

CategoryJavaScript CategoryModes