Installing

Make sure you install the font.

Enabling

There are a few options when it comes down to using ligatures in Emacs. They are listed in order of preferred to less-preferred. Pick one!

- Using composition mode in Emacs Mac port
- Using ligature.el
- Using prettify-symbols
- Using composition char table
- Using font-lock keywords

Using composition mode in Emacs Mac port

If you're using the latest Mac port of Emacs (by Mitsuharu Yamamoto) for macOS, you can use:

```
(mac-auto-operator-composition-mode)
```

Using ligature.el

If you're using a modern version of emacs, you might have ligature support built-in, via HARFBUZZ / Cairo. The ligature.el package adds infrastructure that uses Harfbuzz and Cairo to render ligatures from the fonts, just like in any other editor. Add the following to your `init.el` file:

```
;; Enable the www ligature in every possible major mode
(ligature-set-ligatures 't '("www"))

;; Enable ligatures in programming modes
(ligature-set-ligatures 'prog-mode '("www" "**" "***" "**/" "*>" "*/" "\\\\" "\\\\\\" "{-" "::"
                                     ":::" ":=" "!!" "!=" "!==" "-}" "----" "-->" "->" "->>"
                                     "-<" "-<<" "-~" "#{" "#[" "##" "###" "####" "#(" "#?" "#_"
                                     "#_(" ".-" ".=" ".." "..<" "..." "?=" "??" ";;" "/*" "/**"
                                     "/=" "/==" "/>" "//" "///" "&&" "||" "||=" "|=" "|>" "^=" "$>"
                                     "++" "+++" "+>" "=:=" "==" "===" "==>" "=>" "=>>" "<="
                                     "=<<" "=/=" ">-" ">=" ">=>" ">>" ">>-" ">>=" ">>>" "<*"
                                     "<*>" "<|" "<|>" "<$" "<$>" "<!--" "<-" "<--" "<->" "<+"
                                     "<+>" "<=" "<==" "<=>" "<=<" "<>" "<<" "<<-" "<<=" "<<<"
                                     "<~" "<~~" "</" "</>" "~@" "~-" "~>" "~~" "~~>" "%%"))
```

```
(global-ligature-mode 't)
```

This is generally the easiest solution, ~~but can only be used on macOS~~. Also tested on Linux, it works.

One thing to be aware of is that not all modes where you want code ligatures may derive from `prog-mode`, so you may also want to call `ligature-set-ligatures` with other modes (e.g. `html-mode`).

Using prettify-symbols

Note: fira-code-mode is a MELPA package implementing something similar to this solution, meaning you can implement the below by installing the Fira Code Symbol font and by using the following snippet:

```
(use-package fira-code-mode
  :custom (fira-code-mode-disabled-ligatures '("[]" "x"))  ; ligatures you don't want
  :hook prog-mode)                                         ; mode to enable fira-code-mode in
```

These instructions are pieced together by @Triavanicus, taking some pieces from: Hasklig-Mode.

This method requires you to install the Fira Code Symbol font, made by @siegebell: https://github.com/tonsky/FiraCode/issues/211#issuecomment-239058632

```
(defun fira-code-mode--make-alist (list)
  "Generate prettify-symbols alist from LIST."
  (let ((idx -1))
    (mapcar
     (lambda (s)
       (setq idx (1+ idx))
       (let* ((code (+ #Xe100 idx))
          (width (string-width s))
          (prefix ())
          (suffix '(?\s (Br . Br)))
          (n 1))
     (while (< n width)
       (setq prefix (append prefix '(?\s (Br . Bl))))
       (setq n (1+ n)))
     (cons s (append prefix suffix (list (decode-char 'ucs code))))))
     list)))

(defconst fira-code-mode--ligatures
  '("www" "**" "***" "**/" "*>" "*/" "\\\\" "\\\\\\"
    "{-" "[]" "::" ":::" ":=" "!!" "!=" "!==" "-}"
    "--" "---" "-->" "->" "->>" "-<" "-<<" "-~"
    "#{" "#[" "##" "###" "####" "#(" "#?" "#_" "#_("
    ".-" ".=" ".." "..<" "..." "?=" "??" ";;" "/*"
    "/**" "/=" "/==" "/>" "//" "///" "&&" "||" "||="
    "|=" "|>" "^=" "$>" "++" "+++" "+>" "=:=" "=="
    "===" "==>" "=>" "=>>" "<=" "=<<" "=/=" ">-" ">="
    ">=>" ">>" ">>-" ">>=" ">>>" "<*" "<*>" "<|" "<|>"
    "<$" "<$>" "<!--" "<-" "<--" "<->" "<+" "<+>" "<="
    "<==" "<=>" "<=<" "<>" "<<" "<<-" "<<=" "<<<" "<~"
    "<~~" "</" "</>" "~@" "~-" "~=" "~>" "~~" "~~>" "%%"
    "x" ":" "+" "+" "*"))

(defvar fira-code-mode--old-prettify-alist)

(defun fira-code-mode--enable ()
  "Enable Fira Code ligatures in current buffer."
  (setq-local fira-code-mode--old-prettify-alist prettify-symbols-alist)
  (setq-local prettify-symbols-alist (append (fira-code-mode--make-alist fira-code-mode--ligatures) fira-code-mode--old-prettify-alist))
  (prettify-symbols-mode t))

(defun fira-code-mode--disable ()
  "Disable Fira Code ligatures in current buffer."
  (setq-local prettify-symbols-alist fira-code-mode--old-prettify-alist)
  (prettify-symbols-mode -1))

(define-minor-mode fira-code-mode
  "Fira Code ligatures minor mode"
  :lighter " Fira Code"
  (setq-local prettify-symbols-unprettify-at-point 'right-edge)
  (if fira-code-mode
      (fira-code-mode--enable)
    (fira-code-mode--disable)))

(defun fira-code-mode--setup ()
  "Setup Fira Code Symbols"
  (set-fontset-font t '(#Xe100 . #Xe16f) "Fira Code Symbol"))

(provide 'fira-code-mode)
```

Alternative instructions

https://github.com/Profpatsch/blog/blob/master/posts/ligature-emulation-in-emacs/post.md#appendix-b-update-1-firacode-integration

Using composition char table

Put this lisp in your `.emacs`.

Thanks to Sean Farley for putting this together; extended by Jason Blevins.

```
(when (window-system)
  (set-frame-font "Fira Code"))
(let ((alist '((33 . ".\\(?:\\(?:==\\|!!\\)\\|[!=]\\)")
               (35 . ".\\(?:###\\|##\\|_(\\|[#(?[_{]\\)")
               (36 . ".\\(?:>\\)")
               (37 . ".\\(?:\\(?:%%\\)\\|%\\)")
               (38 . ".\\(?:\\(?:&&\\)\\|&\\)")
               (42 . ".\\(?:\\(?:\\*\\*/\\)\\|\\(?:\\*[*/]\\)\\|[*/>]\\)")
               (43 . ".\\(?:\\(?:\\+\\+\\)\\|[+>]\\)")
               (45 . ".\\(?:\\(?:-[>-]\\|<<\\|>>\\)\\|[<>}~-]\\)")
               (46 . ".\\(?:\\(?:\\.[.<]\\)\\|[.=-]\\)")
               (47 . ".\\(?:\\(?:\\*\\*\\|//\\|==\\)\\|[*/=>]\\)")
               (48 . ".\\(?:x[a-zA-Z]\\)")
               (58 . ".\\(?:::\\|[:=]\\)")
               (59 . ".\\(?:;;\\|;\\)")
               (60 . ".\\(?:\\(?:!--\\)\\|\\(?:~~\\|->\\|\\$>\\|\\*>\\|\\+>\\|--\\|
<[<=-]\\|=[<=>]\\||>\\)\\|[*$+~/<=>|-]\\)")
               (61 . ".\\(?:\\(?:/=\\|:=\\|<<\\|=[=>]\\|>>\\)\\|[<=>~]\\)")
               (62 . ".\\(?:\\(?:=>\\|>[=>-]\\)\\|[=>-]\\)")
               (63 . ".\\(?:\\(\\?\\?\\)\\|[:=?]\\)")
               (91 . ".\\(?:]\\)")
               (92 . ".\\(?:\\(?:\\\\\\\\\\)\\|\\\\\\)")
               (94 . ".\\(?:=\\)")
               (119 . ".\\(?:ww\\)")
               (123 . ".\\(?:-\\)")
               (124 . ".\\(?:\\(?:|[=|]\\)\\|[=>|]\\)")
               (126 . ".\\(?:~>\\|~~\\|[>=@~-]\\)")
               )
             ))
  (dolist (char-regexp alist)
    (set-char-table-range composition-function-table (car char-regexp)
                          `([,(cdr char-regexp) 0 font-shape-gstring]))))
```

Note! If you get `error in process filter: Attempt to shape unibyte text`, check out this issue. Emacs Cider users may avoid this issue by commenting the following line from the above config:

```
;; (46 . ".\\(?:\\(?:\\.[.<]\\)\\|[.=-]\\)")
```

Char `45` is also known to have issues in macOS Mojave.

If you are having problems with helm you can disable ligatures in helm or disable char **46**:

```
   (add-hook 'helm-major-mode-hook
          (lambda ()
            (setq auto-composition-mode nil)))
```

If you are having issues with ediff, you can disable char **45** or disable ligatures in ediff completely:

```
  (add-hook 'ediff-mode-hook
            (lambda ()
              (setq auto-composition-mode nil)))
```

Note! Disabling ligatures in ediff mode only removes them in the ediff buffer itself(the small buffer underneath) and not the buffers you compare. Which is probably a preferred solution

Using font-lock keywords

If none of the above worked, you can try this method.

This method requires you to install the Fira Code Symbol font, made by @siegebell: https://github.com/tonsky/FiraCode/issues/211#issuecomment-239058632

```
;;; Fira code
;; This works when using emacs --daemon + emacsclient
(add-hook 'after-make-frame-functions (lambda (frame) (set-fontset-font t '(#Xe100 .
#Xe16f) "Fira Code Symbol")))
;; This works when using emacs without server/client
(set-fontset-font t '(#Xe100 . #Xe16f) "Fira Code Symbol")
;; I haven't found one statement that makes both of the above situations work, so I
use both for now

(defconst fira-code-font-lock-keywords-alist
  (mapcar (lambda (regex-char-pair)
            `(,(car regex-char-pair)
              (0 (prog1 ()
                   (compose-region (match-beginning 1)
                                   (match-end 1)
                                   ;; The first argument to concat is a string
containing a literal tab
                                   ,(concat "   " (list (decode-char 'ucs (cadr
regex-char-pair)))))))))
          '(("\\(www\\)"                   #Xe100)
            ("[^/]\\(\\*\\*\\)[^/]"        #Xe101)
            ("\\(\\*\\*\\*\\)"             #Xe102)
            ("\\(\\*\\*/\\)"               #Xe103)
            ("\\(\\*>\\)"                  #Xe104)
            ("[^*]\\(\\*/\\)"              #Xe105)
            ("\\(\\\\\\\\\\)"              #Xe106)
            ("\\(\\\\\\\\\\\\\\)"          #Xe107)
            ("\\({-\\)"                    #Xe108)
            ("\\(\\[\\]\\)"                #Xe109)
            ("\\(::\\)"                    #Xe10a)
            ("\\(:::\\)"                   #Xe10b)
            ("[^=]\\(:=\\)"                #Xe10c)
            ("\\(!!\\)"                    #Xe10d)
            ("\\(!=\\)"                    #Xe10e)
            ("\\(!==\\)"                   #Xe10f)
            ("\\(-}\\)"                    #Xe110)
            ("\\(--\\)"                    #Xe111)
            ("\\(---\\)"                   #Xe112)
            ("\\(-->\\)"                   #Xe113)
            ("[^-]\\(->\\)"                #Xe114)
            ("\\(->>\\)"                   #Xe115)
            ("\\(-<\\)"                    #Xe116)
            ("\\(-<<\\)"                   #Xe117)
            ("\\(-~\\)"                    #Xe118)
            ("\\(#{\\)"                    #Xe119)
            ("\\(#\\[\\)"                  #Xe11a)
            ("\\(##\\)"                    #Xe11b)
            ("\\(###\\)"                   #Xe11c)
```

```
("\\(####\\)"                   #Xe11d)
("\\(#(\\)"                     #Xe11e)
("\\(#\\?\\)"                   #Xe11f)
("\\(#_\\)"                     #Xe120)
("\\(#_(\\)"                    #Xe121)
("\\(\\.-\\)"                   #Xe122)
("\\(\\.=\\)"                   #Xe123)
("\\(\\.\\.\\)"                 #Xe124)
("\\(\\.\\.<\\)"                #Xe125)
("\\(\\.\\.\\.\\)"              #Xe126)
("\\(\\?=\\)"                   #Xe127)
("\\(\\?\\?\\)"                 #Xe128)
("\\(;;\\)"                     #Xe129)
("\\(/\\*\\)"                   #Xe12a)
("\\(/\\*\\*\\)"                #Xe12b)
("\\(/=\\)"                     #Xe12c)
("\\(/==\\)"                    #Xe12d)
("\\(/>\\)"                     #Xe12e)
("\\(//\\)"                     #Xe12f)
("\\(///\\)"                    #Xe130)
("\\(&&\\)"                     #Xe131)
("\\(||\\)"                     #Xe132)
("\\(||=\\)"                    #Xe133)
("[^|]\\(|=\\)"                 #Xe134)
("\\(|>\\)"                     #Xe135)
("\\(\\^=\\)"                   #Xe136)
("\\(\\$>\\)"                   #Xe137)
("\\(\\+\\+\\)"                 #Xe138)
("\\(\\+\\+\\+\\)"              #Xe139)
("\\(\\+>\\)"                   #Xe13a)
("\\(=:=\\)"                    #Xe13b)
("[^!/]\\(==\\)[^>]"            #Xe13c)
("\\(===\\)"                    #Xe13d)
("\\(==>\\)"                    #Xe13e)
("[^=]\\(=>\\)"                 #Xe13f)
("\\(=>>\\)"                    #Xe140)
("\\(<=\\)"                     #Xe141)
("\\(=<<\\)"                    #Xe142)
("\\(=/=\\)"                    #Xe143)
("\\(>-\\)"                     #Xe144)
("\\(>=\\)"                     #Xe145)
("\\(>=>\\)"                    #Xe146)
("[^-=]\\(>>\\)"                #Xe147)
("\\(>>-\\)"                    #Xe148)
("\\(>>=\\)"                    #Xe149)
("\\(>>>\\)"                    #Xe14a)
("\\(<\\*\\)"                   #Xe14b)
("\\(<\\*>\\)"                  #Xe14c)
("\\(<|\\)"                     #Xe14d)
("\\(<|>\\)"                    #Xe14e)
("\\(<\\$\\)"                   #Xe14f)
("\\(<\\$>\\)"                  #Xe150)
```

```
            ("\\(<!--\\)"                  #Xe151)
            ("\\(<-\\)"                    #Xe152)
            ("\\(<--\\)"                   #Xe153)
            ("\\(<->\\)"                   #Xe154)
            ("\\(<\\+\\)"                  #Xe155)
            ("\\(<\\+>\\)"                 #Xe156)
            ("\\(<=\\)"                    #Xe157)
            ("\\(<==\\)"                   #Xe158)
            ("\\(<=>\\)"                   #Xe159)
            ("\\(<=<\\)"                   #Xe15a)
            ("\\(<>\\)"                    #Xe15b)
            ("[^-=]\\(<<\\)"               #Xe15c)
            ("\\(<<-\\)"                   #Xe15d)
            ("\\(<<=\\)"                   #Xe15e)
            ("\\(<<<\\)"                   #Xe15f)
            ("\\(<~\\)"                    #Xe160)
            ("\\(<~~\\)"                   #Xe161)
            ("\\(</\\)"                    #Xe162)
            ("\\(</>\\)"                   #Xe163)
            ("\\(~@\\)"                    #Xe164)
            ("\\(~-\\)"                    #Xe165)
            ("\\(~=\\)"                    #Xe166)
            ("\\(~>\\)"                    #Xe167)
            ("[^<]\\(~~\\)"                #Xe168)
            ("\\(~~>\\)"                   #Xe169)
            ("\\(%%\\)"                    #Xe16a)
            ("[0\[]\\(x\\)"                #Xe16b)
            ("[^:=]\\(:\\)[^:=]"           #Xe16c)
            ("[^\\+<>]\\(\\+\\)[^\\+<>]"   #Xe16d)
            ("[^\\*/<>]\\(\\*\\)[^\\*/<>]" #Xe16f))))

(defun add-fira-code-symbol-keywords ()
  (font-lock-add-keywords nil fira-code-font-lock-keywords-alist))

(add-hook 'prog-mode-hook
          #'add-fira-code-symbol-keywords)
```

On some systems, `==` will appear incorrectly as a blank space in certain modes unless you add the following lines to your init file:

```
(set-language-environment "UTF-8")
(set-default-coding-systems 'utf-8)
```