

Emacs Lisp Types

Main Type Category	Sub-category	Sub-category	Sub-category	References
Symbols	Symbols			<ul style="list-style-type: none"> Symbol Type Symbol components Creating and internng symbols
	booleans			
	comparison			<ul style="list-style-type: none"> Equality predicates Comparison functions @ Emacs Wiki
Numbers	Integers			
	Floats			
	complex-numbers			Complex numbers are not explicitly supported by Emacs Lisp. <ul style="list-style-type: none"> The cplx external library supports the concept of complex numbers.
Characters and Strings	Char			<ul style="list-style-type: none"> Basic char literal syntax
	Strings			<ul style="list-style-type: none"> Creating Strings String Modifications @ Emacs Wiki Replace in String @ Emacs Wiki Split String @ Emacs Wiki String trim @ ΣXah
<ul style="list-style-type: none"> Encodings 				<ul style="list-style-type: none"> Unicode Encoding @ Emacs Wiki Emacs Unicode Pitfalls, by Christopher Wellons, 2014-06-13
Ordered Collection of elements See also: <ul style="list-style-type: none"> Pure storage Elisp Cookbook @ Emacs Wiki 	Sequence			<ul style="list-style-type: none"> Sequence Functions
		List		<ul style="list-style-type: none"> List Modifications @ Emacs Wiki List Destructive Operations @ Emacs Wiki Sorting Lists
			cons cell	Note: a cons cell is not a sequence. The cons cell is placed here because of its strong relationship with lists.
			alist - association list	<ul style="list-style-type: none"> Elisp: Association List @ ΣXah
			cl association list	
			plist - property list	<ul style="list-style-type: none"> Elisp: Property List @ ΣXah Alist Vs Plist @ Emacs Wiki
			set - (lists as sets)	
		Array		<ul style="list-style-type: none"> Array Functions
			Vector	<ul style="list-style-type: none"> Vector Functions
			Bool-vector	
			Char-table	
			Ring	
			String	See strings above. Strings are sequences of characters.
Creation of New Object Types	Record			Used as underlying representations of <i>cl-defstruct</i> and <i>defclass</i> instances.
	Structure			<ul style="list-style-type: none"> Options for Structured Data in Emacs Lisp, by Christopher Wellons, 2018-02-14 The Common Lisp Cookbook - Data Structures - Structures
	Hash-table			
ieieo - CLOS for Emacs Lisp	classes			
Emacs Specialized Types	Buffers			
	Markers			<ul style="list-style-type: none"> Functions that create markers
	Overlays			
Functions				<ul style="list-style-type: none"> Emacs Lisp Code Guidelines - Functions, lambdas, macros
	functions			Defining a function with: <ul style="list-style-type: none"> defun macro <ul style="list-style-type: none"> function argument list of functions defined with the defun macro inline functions Calling functions indirectly cl-lib macros: cl-defun, cl-function, cl-defsubst, ... <ul style="list-style-type: none"> See also Common Lisp references (which explain what Emacs Lisp cl-lib emulates): <ul style="list-style-type: none"> The Common Lisp Cookbook – Functions Practical Common Lisp - Common Lisp functions Note that Emacs Lisp cl-defun support <i>&aux auxiliary variables</i> in argument list, something not available in Common Lisp.
	advices <ul style="list-style-type: none"> Ways to compose advice 			<ul style="list-style-type: none"> Advice @ Wikipedia The Limits of Emacs Advice, by Christopher Wellons, 2013-01-22 Advising Functions @ Emacs Wiki Advice Vs Hooks @ Emacs Wiki Meta-Programming in Emacs Using defadvice, T.V. Raman, 2019-10-16
	lambda			<ul style="list-style-type: none"> Anonymous functions lambda expressions What's in an Emacs lambda, by Christopher Wellons, 2017-12-14 Emacs Lisp Lambda Expressions Are Not Self-Evaluating, by Christopher Wellons, 2018-02-22
	closures (lambda)			<ul style="list-style-type: none"> Emacs Lisp Readable Closures, by Christopher Wellons, 2013-12-30 Lexical binding
	generators (and iterators)			<ul style="list-style-type: none"> Emacs 25.1 bring Generators, by Christopher Wellons, 2018-05-31
ieieo - CLOS for Emacs Lisp	methods			<ul style="list-style-type: none"> Generic functions