

Context Stack

Bexp>>ensureWithOnDo ^[[Error signal] ensure: [1]. ^3] on: Error do: [2]	context 1 <i>Bexp new</i>
BlockClosure>>on: exception do: handlerAction handlerActive <primitive: 199> handlerActive := true. ^self value	context 2 [[Error signal] ensure: [1].^3]
BlockClosure>>ensure: aBlock complete returnValue <primitive: 198> returnValue := self valueNoContextSwitch. complete ifNil: [complete := true. aBlock value.]. ^ returnValue	context 3 [Error signal]
Exception class>>signal signalContext := thisContext contextTag. signaler ifNil: [signaler := self receiver]. ^ signalContext <i>nextHandlerContext</i> handleSignal: self	context 4 <i>Error</i>
ContextPart>>handleSignal: exception val ((self exceptionClass handles: exception) and: [self exceptionHandlerIsActive]) ifFalse: [^ self nextHandlerContext handleSignal: exception]. exception privHandlerContext: self contextTag. self exceptionHandlerIsActive: false. val := [self exceptionHandlerBlock cull: exception] ensure: [self exceptionHandlerIsActive: true]. self return: val.	context 5 <i>context 2</i>
ContextPart>>return: value sender ifNil: [self cannotReturn: value to: sender]. sender resume: value	context 6 <i>context 2</i>
ContextPart>>resume: value ctxt unwindBlock self isDead ifTrue: [self cannotReturn: value to: self]. ctxt := thisContext. [ctxt := ctxt <i>findNextUnwindContextUpTo: self.</i> ctxt isNil] whileFalse: [(ctxt tempAt: 2) ifNil:[ctxt tempAt: 2 put: true. unwindBlock := ctxt tempAt: 1. thisContext terminateTo: ctxt. unwindBlock value]]. thisContext <i>terminateTo: self.</i> ^ value	context 7 <i>context 1</i>

