Table of expression forms

Expression	Meaning	Associativity	Argument(s)	Result
`a[...]`	array access (Section 8.1)		`t[]`, integer	`t`
`o.f`	field access (Section 11.8)		object	
`o.m(...)`	method call (Section 11.10)		object	
`x++`	postincrement		numeric	numeric
`x--`	postdecrement		numeric	numeric
`++x`	preincrement		numeric	numeric
`--x`	predecrement		numeric	numeric
`-x`	negation (minus sign)	right	numeric	numeric
`~e`	bitwise complement	right	integer	`int/long`
`!e`	logical negation	right	boolean	boolean
`new t[...]`	array creation (Section 8.1)		type	`t[]`
`new C(...)`	object creation (Section 11.6)		class	`C`
`(t)e`	type cast (Section 11.11)		type, any	`t`
`e1 * e2`	multiplication	left	numeric	numeric
`e1 / e2`	division	left	numeric	numeric
`e1 % e2`	remainder	left	numeric	numeric
`e1 + e2`	addition	left	numeric	numeric
`e1 + e2`	string concatenation	left	String, any	String
`e1 + e2`	string concatenation	left	any, String	String
`e1 - e2`	subtraction	left	numeric	numeric
`e1 << e2`	left shift (Section 11.3)	left	integer	`int/long`
`e1 >> e2`	signed right shift	left	integer	`int/long`
`e1 >>> e2`	unsigned right shift	left	integer	`int/long`
`e1 < e2`	less than	none	numeric	boolean
`e1 <= e2`	less than or equal to	none	numeric	boolean
`e1 >= e2`	greater than or equal to	none	numeric	boolean
`e1 > e2`	greater than	none	numeric	boolean
`e instanceof t`	instance test (Section 11.7)	none	any, ref. type	boolean
`e1 == e2`	equal	left	compatible	boolean
`e1 != e2`	not equal	left	compatible	boolean
`e1 & e2`	bitwise and	left	integer	`int/long`
`e1 & e2`	logical strict and	left	boolean	boolean
`e1 ^ e2`	bitwise exclusive-or	left	integer	`int/long`
`e1 ^ e2`	logical strict exclusive-or	left	boolean	boolean
`e1 | e2`	bitwise or	left	integer	`int/long`
`e1 | e2`	logical strict or	left	boolean	boolean
`e1 && e2`	logical and (Section 11.2)	left	boolean	boolean
`e1 || e2`	logical or (Section 11.2)	left	boolean	boolean
`e1 ? e2 : e3`	conditional (Section 11.5)	right	boolean, any, any	any
`x = e`	assignment (Section 11.4)	right	`e` subtype of `x`	type of `x`
`x += e`	compound assignment	right	compatible	type of `x`