4.15 Binary Operators

The subsections that follow specify the compile-time processing rules of the binary operators. In general, if the operands of a binary operator do not meet the stated requirements, a compile-time error occurs and the result of the operation defaults to type any in further processing. Tables that summarize the compile-time processing rules for operands of the Any type, the Boolean, Number, and String primitive types, and all object types and type parameters (the Object column in the tables) are provided.

4.15.1 The *, /, %, –, <<, >>, >>>, &, ^, and | operators

These operators require their operands to be of type Any, the Number primitive type, or an enum type. Operands of an enum type are treated as having the primitive type Number. If one operand is the `null` or `undefined` value, it is treated as having the type of the other operand. The result is always of the Number primitive type.

	Any	Boolean	Number	String	Object
Any	Number		Number		
Boolean					
Number	Number		Number		
String					
Object					

4.15.2 The + operator

The binary + operator requires both operands to be of the Number primitive type or an enum type, or at least one of the operands to be of type Any or the String primitive type. Operands of an enum type are treated as having the primitive type Number. If one operand is the `null` or `undefined` value, it is treated as having the type of the other operand. If both operands are of the Number primitive type, the result is of the Number primitive type. If one or both operands are of the String primitive type, the result is of the String primitive type. Otherwise, the result is of type Any.

	Any	Boolean	Number	String	Object
Any	Any	Any	Any	String	Any
Boolean	Any			String	
Number	Any		Number	String	
String	String	String	String	String	String
Object	Any			String	

A value of any type can converted to the String primitive type by adding an empty string: