ON THIS PAGE

Summary of the Grammar

Lexical Structure

GRAMMAR OF WHITESPACE

whitespace → whitespace-item $whitespace_{opt}$
whitespace-item → line-break
whitespace-item → comment
whitespace-item → multiline-comment
whitespace-item → U+0000, U+0009, U+000B, U+000C, or U+0020

line-break → U+000A
line-break → U+000D
line-break → U+000D followed by U+000A

comment → **//** comment-text line-break
multiline-comment → **/*** multiline-comment-text ***/**

comment-text → comment-text-item $comment\text{-}text_{opt}$
comment-text-item → Any Unicode scalar value except U+000A or U+000D

multiline-comment-text → multiline-comment-text-item $multiline\text{-}comment\text{-}text_{opt}$
multiline-comment-text-item → multiline-comment
multiline-comment-text-item → comment-text-item
multiline-comment-text-item → Any Unicode scalar value except **/*** or ***/**

GRAMMAR OF AN IDENTIFIER

identifier → identifier-head $identifier\text{-}characters_{opt}$
identifier → **`** identifier-head $identifier\text{-}characters_{opt}$ **`**
identifier → implicit-parameter-name

identifier-list → identifier | identifier **,** identifier-list

identifier-head → Upper- or lowercase letter A through Z
identifier-head → **_**
identifier-head → U+00A8, U+00AA, U+00AD, U+00AF, U+00B2–U+00B5, or U+00B7–U+00BA
identifier-head → U+00BC–U+00BE, U+00C0–U+00D6, U+00D8–U+00F6, or U+00F8–U+00FF
identifier-head → U+0100–U+02FF, U+0370–U+167F, U+1681–U+180D, or U+180F–U+1DBF
identifier-head → U+1E00–U+1FFF
identifier-head → U+200B–U+200D, U+202A–U+202E, U+203F–U+2040, U+2054, or U+2060–U+206F
identifier-head → U+2070–U+20CF, U+2100–U+218F, U+2460–U+24FF, or U+2776–U+2793
identifier-head → U+2C00–U+2DFF or U+2E80–U+2FFF
identifier-head → U+3004–U+3007, U+3021–U+302F, U+3031–U+303F, or U+3040–U+D7FF
identifier-head → U+F900–U+FD3D, U+FD40–U+FDCF, U+FDF0–U+FE1F, or U+FE30–U+FE44
identifier-head → U+FE47–U+FFFD
identifier-head → U+10000–U+1FFFD, U+20000–U+2FFFD, U+30000–U+3FFFD, or U+40000–U+4FFFD
identifier-head → U+50000–U+5FFFD, U+60000–U+6FFFD, U+70000–U+7FFFD, or U+80000–U+8FFFD
identifier-head → U+90000–U+9FFFD, U+A0000–U+AFFFD, U+B0000–U+BFFFD, or U+C0000–U+CFFFD
identifier-head → U+D0000–U+DFFFD or U+E0000–U+EFFFD

identifier-character → Digit 0 through 9
identifier-character → U+0300–U+036F, U+1DC0–U+1DFF, U+20D0–U+20FF, or U+FE20–U+FE2F
identifier-character → identifier-head
identifier-characters → identifier-character identifier-characters$_{opt}$

implicit-parameter-name → **$** decimal-digits

GRAMMAR OF A LITERAL

literal → numeric-literal | string-literal | boolean-literal | nil-literal

numeric-literal → **–**$_{opt}$ integer-literal | **–**$_{opt}$ floating-point-literal
boolean-literal → **true** | **false**
nil-literal → **nil**

GRAMMAR OF AN INTEGER LITERAL

integer-literal → binary-literal
integer-literal → octal-literal
integer-literal → decimal-literal
integer-literal → hexadecimal-literal

binary-literal → **0b** binary-digit binary-literal-characters$_{opt}$
binary-digit → Digit 0 or 1
binary-literal-character → binary-digit | **_**
binary-literal-characters → binary-literal-character binary-literal-characters$_{opt}$

octal-literal → **0o** octal-digit octal-literal-characters$_{opt}$
octal-digit → Digit 0 through 7
octal-literal-character → octal-digit | **_**
octal-literal-characters → octal-literal-character octal-literal-characters$_{opt}$

decimal-literal → decimal-digit decimal-literal-characters$_{opt}$
decimal-digit → Digit 0 through 9
decimal-digits → decimal-digit decimal-digits$_{opt}$
decimal-literal-character → decimal-digit | **_**
decimal-literal-characters → decimal-literal-character decimal-literal-characters$_{opt}$

hexadecimal-literal → **0x** hexadecimal-digit hexadecimal-literal-characters$_{opt}$
hexadecimal-digit → Digit 0 through 9, a through f, or A through F
hexadecimal-literal-character → hexadecimal-digit | **_**
hexadecimal-literal-characters → hexadecimal-literal-character hexadecimal-literal-characters$_{opt}$

GRAMMAR OF A FLOATING-POINT LITERAL

floating-point-literal → decimal-literal decimal-fraction$_{opt}$ decimal-exponent$_{opt}$
floating-point-literal → hexadecimal-literal hexadecimal-fraction$_{opt}$ hexadecimal-exponent

decimal-fraction → **.** decimal-literal
decimal-exponent → floating-point-e sign$_{opt}$ decimal-literal

hexadecimal-fraction → **.** hexadecimal-digit hexadecimal-literal-characters$_{opt}$
hexadecimal-exponent → floating-point-p sign$_{opt}$ decimal-literal

floating-point-e → **e** | **E**
floating-point-p → **p** | **P**
sign → **+** | **–**

GRAMMAR OF A STRING LITERAL

string-literal → static-string-literal | interpolated-string-literal

string-literal-opening-delimiter → extended-string-literal-delimiter$_{opt}$ **"**
string-literal-closing-delimiter → **"** extended-string-literal-delimiter$_{opt}$

static-string-literal → string-literal-opening-delimiter quoted-text$_{opt}$ string-literal-closing-delimiter
static-string-literal → multiline-string-literal-opening-delimiter multiline-quoted-text$_{opt}$ multiline-string-literal-closing-delimiter

multiline-string-literal-opening-delimiter → extended-string-literal-delimiter **"""**
multiline-string-literal-closing-delimiter → **"""** extended-string-literal-delimiter
extended-string-literal-delimiter → **#** extended-string-literal-delimiter$_{opt}$

quoted-text → quoted-text-item quoted-text$_{opt}$
quoted-text-item → escaped-character
quoted-text-item → Any Unicode scalar value except **"**, ****, U+000A, or U+000D

multiline-quoted-text → multiline-quoted-text-item multiline-quoted-text$_{opt}$
multiline-quoted-text-item → escaped-character
multiline-quoted-text-item → Any Unicode scalar value except ****
multiline-quoted-text-item → escaped-newline

interpolated-string-literal → string-literal-opening-delimiter interpolated-text$_{opt}$ string-literal-closing-delimiter
interpolated-string-literal → multiline-string-literal-opening-delimiter interpolated-text$_{opt}$ multiline-string-literal-closing-delimiter

interpolated-text → interpolated-text-item interpolated-text$_{opt}$
interpolated-text-item → **\(** expression **)** | quoted-text-item

multiline-interpolated-text → multiline-interpolated-text-item multiline-interpolated-text$_{opt}$
multiline-interpolated-text-item → **\(** expression **)** | multiline-quoted-text-item

escape-sequence → **** extended-string-literal-delimiter
escaped-character → escape-sequence **0** | escape-sequence **** | escape-sequence **t** | escape-sequence **n** | escape-sequence **r** | escape-sequence **"** | escape-sequence **'**
escaped-character → escape-sequence **u** **{** unicode-scalar-digits **}**
unicode-scalar-digits → Between one and eight hexadecimal digits

escaped-newline → escape-sequence whitespace$_{opt}$ line-break

GRAMMAR OF OPERATORS

operator → operator-head operator-characters$_{opt}$
operator → dot-operator-head dot-operator-characters

operator-head → **/** | **=** | **-** | **+** | **!** | ***** | **%** | **<** | **>** | **&** | **|** | **^** | **~** | **?**
operator-head → U+00A1–U+00A7

operator-head → U+00A9 or U+00AB
operator-head → U+00AC or U+00AE
operator-head → U+00B0–U+00B1
operator-head → U+00B6, U+00BB, U+00BF, U+00D7, or U+00F7
operator-head → U+2016–U+2017
operator-head → U+2020–U+2027
operator-head → U+2030–U+203E
operator-head → U+2041–U+2053
operator-head → U+2055–U+205E
operator-head → U+2190–U+23FF
operator-head → U+2500–U+2775
operator-head → U+2794–U+2BFF
operator-head → U+2E00–U+2E7F
operator-head → U+3001–U+3003
operator-head → U+3008–U+3020
operator-head → U+3030

operator-character → operator-head
operator-character → U+0300–U+036F
operator-character → U+1DC0–U+1DFF
operator-character → U+20D0–U+20FF
operator-character → U+FE00–U+FE0F
operator-character → U+FE20–U+FE2F
operator-character → U+E0100–U+E01EF
operator-characters → operator-character operator-characters$_{opt}$

dot-operator-head → **.**
dot-operator-character → **.** | operator-character
dot-operator-characters → dot-operator-character dot-operator-characters$_{opt}$

binary-operator → operator
prefix-operator → operator
postfix-operator → operator

Types

GRAMMAR OF A TYPE

type → function-type
type → array-type
type → dictionary-type

type → type-identifier
type → tuple-type
type → optional-type
type → implicitly-unwrapped-optional-type
type → protocol-composition-type
type → opaque-type
type → metatype-type
type → self-type
type → **Any**
type → **(** type **)**

GRAMMAR OF A TYPE ANNOTATION

type-annotation → **:** attributes$_{opt}$ **inout**$_{opt}$ type

GRAMMAR OF A TYPE IDENTIFIER

type-identifier → type-name generic-argument-clause$_{opt}$ | type-name generic-argument-clause$_{opt}$ **.** type-identifier
type-name → identifier

GRAMMAR OF A TUPLE TYPE

tuple-type → **(** **)** | **(** tuple-type-element **,** tuple-type-element-list **)**
tuple-type-element-list → tuple-type-element | tuple-type-element **,** tuple-type-element-list
tuple-type-element → element-name type-annotation | type
element-name → identifier

GRAMMAR OF A FUNCTION TYPE

function-type → attributes$_{opt}$ function-type-argument-clause **throws**$_{opt}$ **->** type
function-type → attributes$_{opt}$ function-type-argument-clause **rethrows** **->** type

function-type-argument-clause → **(** **)**
function-type-argument-clause → **(** function-type-argument-list **...**$_{opt}$ **)**

function-type-argument-list → function-type-argument | function-type-argument **,** function-type-argument-list
function-type-argument → attributes$_{opt}$ **inout**$_{opt}$ type | argument-label type-annotation
argument-label → identifier

GRAMMAR OF AN ARRAY TYPE

array-type → **[** type **]**

GRAMMAR OF A DICTIONARY TYPE

dictionary-type → **[** type **:** type **]**

GRAMMAR OF AN OPTIONAL TYPE

optional-type → type **?**

GRAMMAR OF AN IMPLICITLY UNWRAPPED OPTIONAL TYPE

implicitly-unwrapped-optional-type → type **!**

GRAMMAR OF A PROTOCOL COMPOSITION TYPE

protocol-composition-type → type-identifier **&** protocol-composition-continuation
protocol-composition-continuation → type-identifier | protocol-composition-type

GRAMMAR OF AN OPAQUE TYPE

opaque-type → **some** type

GRAMMAR OF A METATYPE TYPE

metatype-type → type **. Type** | type **. Protocol**

GRAMMAR OF A SELF TYPE

self-type → **Self**

GRAMMAR OF A TYPE INHERITANCE CLAUSE

type-inheritance-clause → **:** type-inheritance-list
type-inheritance-list → type-identifier | type-identifier **,** type-inheritance-list

Expressions

GRAMMAR OF AN EXPRESSION

expression → try-operator$_{opt}$ prefix-expression binary-expressions$_{opt}$
expression-list → expression | expression **,** expression-list

GRAMMAR OF A PREFIX EXPRESSION

prefix-expression → prefix-operator$_{opt}$ postfix-expression
prefix-expression → in-out-expression
in-out-expression → **&** identifier

GRAMMAR OF A TRY EXPRESSION

try-operator → **try** | **try ?** | **try !**

GRAMMAR OF A BINARY EXPRESSION

binary-expression → binary-operator prefix-expression
binary-expression → assignment-operator try-operator$_{opt}$ prefix-expression
binary-expression → conditional-operator try-operator$_{opt}$ prefix-expression
binary-expression → type-casting-operator
binary-expressions → binary-expression binary-expressions$_{opt}$

GRAMMAR OF AN ASSIGNMENT OPERATOR

assignment-operator → **=**

GRAMMAR OF A CONDITIONAL OPERATOR

conditional-operator → **?** expression **:**

GRAMMAR OF A TYPE-CASTING OPERATOR

type-casting-operator → **is** type
type-casting-operator → **as** type
type-casting-operator → **as ?** type
type-casting-operator → **as !** type

GRAMMAR OF A PRIMARY EXPRESSION

primary-expression → identifier generic-argument-clause$_{opt}$
primary-expression → literal-expression
primary-expression → self-expression

primary-expression → superclass-expression
primary-expression → closure-expression
primary-expression → parenthesized-expression
primary-expression → tuple-expression
primary-expression → implicit-member-expression
primary-expression → wildcard-expression
primary-expression → key-path-expression
primary-expression → selector-expression
primary-expression → key-path-string-expression

GRAMMAR OF A LITERAL EXPRESSION

literal-expression → literal
literal-expression → array-literal | dictionary-literal | playground-literal
literal-expression → **#file** | **#line** | **#column** | **#function** | **#dsohandle**

array-literal → **[** array-literal-items$_{opt}$ **]**
array-literal-items → array-literal-item **,**$_{opt}$ | array-literal-item **,** array-literal-items
array-literal-item → expression

dictionary-literal → **[** dictionary-literal-items **]** | **[:]**
dictionary-literal-items → dictionary-literal-item **,**$_{opt}$ | dictionary-literal-item **,** dictionary-literal-items
dictionary-literal-item → expression **:** expression

playground-literal → **#colorLiteral (red :** expression **, green :** expression **, blue :** expression **, alpha :** expression **)**
playground-literal → **#fileLiteral (resourceName :** expression **)**
playground-literal → **#imageLiteral (resourceName :** expression **)**

GRAMMAR OF A SELF EXPRESSION

self-expression → **self** | self-method-expression | self-subscript-expression | self-initializer-expression

self-method-expression → **self .** identifier
self-subscript-expression → **self [** function-call-argument-list **]**
self-initializer-expression → **self . init**

GRAMMAR OF A SUPERCLASS EXPRESSION

superclass-expression → superclass-method-expression | superclass-subscript-expression | superclass-initializer-expression

superclass-method-expression → **super** **.** identifier
superclass-subscript-expression → **super** **[** function-call-argument-list **]**
superclass-initializer-expression → **super** **.** **init**

GRAMMAR OF A CLOSURE EXPRESSION

closure-expression → **{** closure-signature$_{opt}$ statements$_{opt}$ **}**

closure-signature → capture-list$_{opt}$ closure-parameter-clause **throws**$_{opt}$ function-result$_{opt}$ **in**
closure-signature → capture-list **in**

closure-parameter-clause → **(** **)** | **(** closure-parameter-list **)** | identifier-list
closure-parameter-list → closure-parameter | closure-parameter **,** closure-parameter-list
closure-parameter → closure-parameter-name type-annotation$_{opt}$
closure-parameter → closure-parameter-name type-annotation **...**
closure-parameter-name → identifier

capture-list → **[** capture-list-items **]**
capture-list-items → capture-list-item | capture-list-item **,** capture-list-items
capture-list-item → capture-specifier$_{opt}$ expression
capture-specifier → **weak** | **unowned** | **unowned(safe)** | **unowned(unsafe)**

GRAMMAR OF A IMPLICIT MEMBER EXPRESSION

implicit-member-expression → **.** identifier

GRAMMAR OF A PARENTHESIZED EXPRESSION

parenthesized-expression → **(** expression **)**

GRAMMAR OF A TUPLE EXPRESSION

tuple-expression → **(** **)** | **(** tuple-element **,** tuple-element-list **)**
tuple-element-list → tuple-element | tuple-element **,** tuple-element-list
tuple-element → expression | identifier **:** expression

GRAMMAR OF A WILDCARD EXPRESSION

wildcard-expression → **_**

GRAMMAR OF A KEY-PATH EXPRESSION

key-path-expression → **** type$_{opt}$ **.** key-path-components
key-path-components → key-path-component | key-path-component **.** key-path-components
key-path-component → identifier key-path-postfixes$_{opt}$ | key-path-postfixes

key-path-postfixes → key-path-postfix key-path-postfixes$_{opt}$
key-path-postfix → **?** | **!** | **self** | **[** function-call-argument-list **]**

GRAMMAR OF A SELECTOR EXPRESSION

selector-expression → **#selector (** expression **)**
selector-expression → **#selector (getter:** expression **)**
selector-expression → **#selector (setter:** expression **)**

GRAMMAR OF A KEY-PATH STRING EXPRESSION

key-path-string-expression → **#keyPath (** expression **)**

GRAMMAR OF A POSTFIX EXPRESSION

postfix-expression → primary-expression
postfix-expression → postfix-expression postfix-operator
postfix-expression → function-call-expression
postfix-expression → initializer-expression
postfix-expression → explicit-member-expression
postfix-expression → postfix-self-expression
postfix-expression → subscript-expression
postfix-expression → forced-value-expression
postfix-expression → optional-chaining-expression

GRAMMAR OF A FUNCTION CALL EXPRESSION

function-call-expression → postfix-expression function-call-argument-clause
function-call-expression → postfix-expression function-call-argument-clause$_{opt}$ trailing-closure

function-call-argument-clause → **( )** | **(** function-call-argument-list **)**
function-call-argument-list → function-call-argument | function-call-argument **,** function-call-argument-list
function-call-argument → expression | identifier **:** expression
function-call-argument → operator | identifier **:** operator

trailing-closure → closure-expression

GRAMMAR OF AN INITIALIZER EXPRESSION

initializer-expression → postfix-expression **.** **init**
initializer-expression → postfix-expression **.** **init** **(** argument-names **)**

GRAMMAR OF AN EXPLICIT MEMBER EXPRESSION

explicit-member-expression → postfix-expression **.** decimal-digits
explicit-member-expression → postfix-expression **.** identifier generic-argument-clause*opt*
explicit-member-expression → postfix-expression **.** identifier **(** argument-names **)**

argument-names → argument-name argument-names*opt*
argument-name → identifier **:**

GRAMMAR OF A POSTFIX SELF EXPRESSION

postfix-self-expression → postfix-expression **.** **self**

GRAMMAR OF A SUBSCRIPT EXPRESSION

subscript-expression → postfix-expression **[** function-call-argument-list **]**

GRAMMAR OF A FORCED-VALUE EXPRESSION

forced-value-expression → postfix-expression **!**

GRAMMAR OF AN OPTIONAL-CHAINING EXPRESSION

optional-chaining-expression → postfix-expression **?**

Statements

GRAMMAR OF A STATEMENT

statement → expression **;***opt*
statement → declaration **;***opt*
statement → loop-statement **;***opt*
statement → branch-statement **;***opt*
statement → labeled-statement **;***opt*
statement → control-transfer-statement **;***opt*
statement → defer-statement **;***opt*

statement → do-statement **;**$_{opt}$
statement → compiler-control-statement
statements → statement statements$_{opt}$

GRAMMAR OF A LOOP STATEMENT

loop-statement → for-in-statement
loop-statement → while-statement
loop-statement → repeat-while-statement

GRAMMAR OF A FOR-IN STATEMENT

for-in-statement → **for** **case**$_{opt}$ pattern **in** expression where-clause$_{opt}$ code-block

GRAMMAR OF A WHILE STATEMENT

while-statement → **while** condition-list code-block

condition-list → condition | condition **,** condition-list
condition → expression | availability-condition | case-condition | optional-binding-condition

case-condition → **case** pattern initializer
optional-binding-condition → **let** pattern initializer | **var** pattern initializer

GRAMMAR OF A REPEAT-WHILE STATEMENT

repeat-while-statement → **repeat** code-block **while** expression

GRAMMAR OF A BRANCH STATEMENT

branch-statement → if-statement
branch-statement → guard-statement
branch-statement → switch-statement

GRAMMAR OF AN IF STATEMENT

if-statement → **if** condition-list code-block else-clause$_{opt}$
else-clause → **else** code-block | **else** if-statement

GRAMMAR OF A GUARD STATEMENT

guard-statement → **guard** condition-list **else** code-block

GRAMMAR OF A SWITCH STATEMENT

switch-statement → **switch** expression **{** switch-cases$_{opt}$ **}**
switch-cases → switch-case switch-cases$_{opt}$
switch-case → case-label statements
switch-case → default-label statements
switch-case → conditional-switch-case

case-label → attributes$_{opt}$ **case** case-item-list **:**
case-item-list → pattern where-clause$_{opt}$ | pattern where-clause$_{opt}$ **,** case-item-list
default-label → attributes$_{opt}$ **default :**

where-clause → **where** where-expression
where-expression → expression

conditional-switch-case → switch-if-directive-clause switch-elseif-directive-clauses$_{opt}$ switch-else-directive-clause$_{opt}$ endif-directive
switch-if-directive-clause → if-directive compilation-condition switch-cases$_{opt}$
switch-elseif-directive-clauses → elseif-directive-clause switch-elseif-directive-clauses$_{opt}$
switch-elseif-directive-clause → elseif-directive compilation-condition switch-cases$_{opt}$
switch-else-directive-clause → else-directive switch-cases$_{opt}$

GRAMMAR OF A LABELED STATEMENT

labeled-statement → statement-label loop-statement
labeled-statement → statement-label if-statement
labeled-statement → statement-label switch-statement
labeled-statement → statement-label do-statement

statement-label → label-name **:**
label-name → identifier

GRAMMAR OF A CONTROL TRANSFER STATEMENT

control-transfer-statement → break-statement
control-transfer-statement → continue-statement
control-transfer-statement → fallthrough-statement
control-transfer-statement → return-statement
control-transfer-statement → throw-statement

GRAMMAR OF A BREAK STATEMENT

break-statement → **break** label-name$_{opt}$

GRAMMAR OF A CONTINUE STATEMENT

continue-statement → **continue** label-name*opt*

GRAMMAR OF A FALLTHROUGH STATEMENT

fallthrough-statement → **fallthrough**

GRAMMAR OF A RETURN STATEMENT

return-statement → **return** expression*opt*

GRAMMAR OF A THROW STATEMENT

throw-statement → **throw** expression

GRAMMAR OF A DEFER STATEMENT

defer-statement → **defer** code-block

GRAMMAR OF A DO STATEMENT

do-statement → **do** code-block catch-clauses*opt*
catch-clauses → catch-clause catch-clauses*opt*
catch-clause → **catch** pattern*opt* where-clause*opt* code-block

GRAMMAR OF A COMPILER CONTROL STATEMENT

compiler-control-statement → conditional-compilation-block
compiler-control-statement → line-control-statement
compiler-control-statement → diagnostic-statement

GRAMMAR OF A CONDITIONAL COMPILATION BLOCK

conditional-compilation-block → if-directive-clause elseif-directive-clauses*opt* else-directive-clause*opt* endif-directive

if-directive-clause → if-directive compilation-condition statements*opt*
elseif-directive-clauses → elseif-directive-clause elseif-directive-clauses*opt*
elseif-directive-clause → elseif-directive compilation-condition statements*opt*
else-directive-clause → else-directive statements*opt*
if-directive → **#if**
elseif-directive → **#elseif**

else-directive → **#else**
endif-directive → **#endif**

compilation-condition → platform-condition
compilation-condition → identifier
compilation-condition → boolean-literal
compilation-condition → **(** compilation-condition **)**
compilation-condition → **!** compilation-condition
compilation-condition → compilation-condition **&&** compilation-condition
compilation-condition → compilation-condition **||** compilation-condition

platform-condition → **os (** operating-system **)**
platform-condition → **arch (** architecture **)**
platform-condition → **swift (>=** swift-version **)** | **swift (<** swift-version **)**
platform-condition → **compiler (>=** swift-version **)** | **compiler (<** swift-version **)**
platform-condition → **canImport (** module-name **)**
platform-condition → **targetEnvironment (** environment **)**

operating-system → **macOS** | **iOS** | **watchOS** | **tvOS**
architecture → **i386** | **x86_64** | **arm** | **arm64**
swift-version → decimal-digits swift-version-continuation*opt*
swift-version-continuation → **.** decimal-digits swift-version-continuation*opt*
module-name → identifier
environment → **simulator**

GRAMMAR OF A LINE CONTROL STATEMENT

line-control-statement → **#sourceLocation (file:** file-name **, line:** line-number **)**
line-control-statement → **#sourceLocation ( )**
line-number → A decimal integer greater than zero
file-name → static-string-literal

GRAMMAR OF A COMPILE-TIME DIAGNOSTIC STATEMENT

diagnostic-statement → **#error (** diagnostic-message **)**
diagnostic-statement → **#warning (** diagnostic-message **)**

diagnostic-message → static-string-literal

GRAMMAR OF AN AVAILABILITY CONDITION

availability-condition → **#available (** availability-arguments **)**
availability-arguments → availability-argument | availability-argument **,** availability-arguments
availability-argument → platform-name platform-version
availability-argument → *****

platform-name → **iOS** | **iOSApplicationExtension**
platform-name → **macOS** | **macOSApplicationExtension**
platform-name → **watchOS**
platform-name → **tvOS**
platform-version → decimal-digits
platform-version → decimal-digits **.** decimal-digits
platform-version → decimal-digits **.** decimal-digits **.** decimal-digits

Declarations

GRAMMAR OF A DECLARATION

declaration → import-declaration
declaration → constant-declaration
declaration → variable-declaration
declaration → typealias-declaration
declaration → function-declaration
declaration → enum-declaration
declaration → struct-declaration
declaration → class-declaration
declaration → protocol-declaration
declaration → initializer-declaration
declaration → deinitializer-declaration
declaration → extension-declaration
declaration → subscript-declaration
declaration → operator-declaration
declaration → precedence-group-declaration
declarations → declaration declarations$_{opt}$

GRAMMAR OF A TOP-LEVEL DECLARATION

top-level-declaration → statements$_{opt}$

GRAMMAR OF A CODE BLOCK

code-block → **{** statements$_{opt}$ **}**

GRAMMAR OF AN IMPORT DECLARATION

import-declaration → attributes$_{opt}$ **import** import-kind$_{opt}$ import-path

import-kind → **typealias** | **struct** | **class** | **enum** | **protocol** | **let** | **var** | **func**

import-path → import-path-identifier | import-path-identifier **.** import-path

import-path-identifier → identifier | operator

GRAMMAR OF A CONSTANT DECLARATION

constant-declaration → attributes$_{opt}$ declaration-modifiers$_{opt}$ **let** pattern-initializer-list

pattern-initializer-list → pattern-initializer | pattern-initializer **,** pattern-initializer-list

pattern-initializer → pattern initializer$_{opt}$

initializer → **=** expression

GRAMMAR OF A VARIABLE DECLARATION

variable-declaration → variable-declaration-head pattern-initializer-list

variable-declaration → variable-declaration-head variable-name type-annotation code-block

variable-declaration → variable-declaration-head variable-name type-annotation getter-setter-block

variable-declaration → variable-declaration-head variable-name type-annotation getter-setter-keyword-block

variable-declaration → variable-declaration-head variable-name initializer willSet-didSet-block

variable-declaration → variable-declaration-head variable-name type-annotation initializer$_{opt}$ willSet-didSet-block

variable-declaration-head → attributes$_{opt}$ declaration-modifiers$_{opt}$ **var**

variable-name → identifier

getter-setter-block → code-block

getter-setter-block → **{** getter-clause setter-clause$_{opt}$ **}**

getter-setter-block → **{** setter-clause getter-clause **}**

getter-clause → attributes$_{opt}$ mutation-modifier$_{opt}$ **get** code-block

setter-clause → attributes$_{opt}$ mutation-modifier$_{opt}$ **set** setter-name$_{opt}$ code-block

setter-name → **(** identifier **)**

getter-setter-keyword-block → **{** getter-keyword-clause setter-keyword-clause$_{opt}$ **}**

getter-setter-keyword-block → **{** setter-keyword-clause getter-keyword-clause **}**
getter-keyword-clause → attributes$_{opt}$ mutation-modifier$_{opt}$ **get**
setter-keyword-clause → attributes$_{opt}$ mutation-modifier$_{opt}$ **set**

willSet-didSet-block → **{** willSet-clause didSet-clause$_{opt}$ **}**
willSet-didSet-block → **{** didSet-clause willSet-clause$_{opt}$ **}**
willSet-clause → attributes$_{opt}$ **willSet** setter-name$_{opt}$ code-block
didSet-clause → attributes$_{opt}$ **didSet** setter-name$_{opt}$ code-block

GRAMMAR OF A TYPE ALIAS DECLARATION

typealias-declaration → attributes$_{opt}$ access-level-modifier$_{opt}$ **typealias** typealias-name generic-parameter-clause$_{opt}$ typealias-assignment
typealias-name → identifier
typealias-assignment → **=** type

GRAMMAR OF A FUNCTION DECLARATION

function-declaration → function-head function-name generic-parameter-clause$_{opt}$ function-signature generic-where-clause$_{opt}$ function-body$_{opt}$

function-head → attributes$_{opt}$ declaration-modifiers$_{opt}$ **func**
function-name → identifier | operator

function-signature → parameter-clause **throws**$_{opt}$ function-result$_{opt}$
function-signature → parameter-clause **rethrows** function-result$_{opt}$
function-result → **->** attributes$_{opt}$ type
function-body → code-block

parameter-clause → **(** **)** | **(** parameter-list **)**
parameter-list → parameter | parameter **,** parameter-list
parameter → external-parameter-name$_{opt}$ local-parameter-name type-annotation default-argument-clause$_{opt}$
parameter → external-parameter-name$_{opt}$ local-parameter-name type-annotation
parameter → external-parameter-name$_{opt}$ local-parameter-name type-annotation **...**
external-parameter-name → identifier
local-parameter-name → identifier
default-argument-clause → **=** expression

GRAMMAR OF AN ENUMERATION DECLARATION

enum-declaration → attributes$_{opt}$ access-level-modifier$_{opt}$ union-style-enum
enum-declaration → attributes$_{opt}$ access-level-modifier$_{opt}$ raw-value-style-enum

union-style-enum → **indirect**$_{opt}$ **enum** enum-name generic-parameter-clause$_{opt}$ type-inheritance-clause$_{opt}$ generic-where-clause$_{opt}$ **{** union-style-enum-members$_{opt}$ **}**
union-style-enum-members → union-style-enum-member union-style-enum-members$_{opt}$
union-style-enum-member → declaration | union-style-enum-case-clause | compiler-control-statement
union-style-enum-case-clause → attributes$_{opt}$ **indirect**$_{opt}$ **case** union-style-enum-case-list
union-style-enum-case-list → union-style-enum-case | union-style-enum-case **,** union-style-enum-case-list
union-style-enum-case → enum-case-name tuple-type$_{opt}$
enum-name → identifier
enum-case-name → identifier

raw-value-style-enum → **enum** enum-name generic-parameter-clause$_{opt}$ type-inheritance-clause generic-where-clause$_{opt}$ **{** raw-value-style-enum-members **}**
raw-value-style-enum-members → raw-value-style-enum-member raw-value-style-enum-members$_{opt}$
raw-value-style-enum-member → declaration | raw-value-style-enum-case-clause | compiler-control-statement
raw-value-style-enum-case-clause → attributes$_{opt}$ **case** raw-value-style-enum-case-list
raw-value-style-enum-case-list → raw-value-style-enum-case | raw-value-style-enum-case **,** raw-value-style-enum-case-list
raw-value-style-enum-case → enum-case-name raw-value-assignment$_{opt}$
raw-value-assignment → **=** raw-value-literal
raw-value-literal → numeric-literal | static-string-literal | boolean-literal

GRAMMAR OF A STRUCTURE DECLARATION

struct-declaration → attributes$_{opt}$ access-level-modifier$_{opt}$ **struct** struct-name generic-parameter-clause$_{opt}$ type-inheritance-clause$_{opt}$ generic-where-clause$_{opt}$ struct-body
struct-name → identifier
struct-body → **{** struct-members$_{opt}$ **}**

struct-members → struct-member struct-members$_{opt}$
struct-member → declaration | compiler-control-statement

GRAMMAR OF A CLASS DECLARATION

class-declaration → attributes$_{opt}$ access-level-modifier$_{opt}$ **final**$_{opt}$ **class** class-name generic-parameter-clause$_{opt}$ type-inheritance-clause$_{opt}$ generic-where-clause$_{opt}$ class-body

class-declaration → attributes$_{opt}$ **final** access-level-modifier$_{opt}$ **class** class-name generic-parameter-clause$_{opt}$ type-inheritance-clause$_{opt}$ generic-where-clause$_{opt}$ class-body
class-name → identifier
class-body → **{** class-members$_{opt}$ **}**

class-members → class-member class-members$_{opt}$
class-member → declaration | compiler-control-statement

GRAMMAR OF A PROTOCOL DECLARATION

protocol-declaration → attributes$_{opt}$ access-level-modifier$_{opt}$ **protocol** protocol-name type-inheritance-clause$_{opt}$ generic-where-clause$_{opt}$ protocol-body
protocol-name → identifier
protocol-body → **{** protocol-members$_{opt}$ **}**

protocol-members → protocol-member protocol-members$_{opt}$
protocol-member → protocol-member-declaration | compiler-control-statement

protocol-member-declaration → protocol-property-declaration
protocol-member-declaration → protocol-method-declaration
protocol-member-declaration → protocol-initializer-declaration
protocol-member-declaration → protocol-subscript-declaration
protocol-member-declaration → protocol-associated-type-declaration
protocol-member-declaration → typealias-declaration

GRAMMAR OF A PROTOCOL PROPERTY DECLARATION

protocol-property-declaration → variable-declaration-head variable-name type-annotation getter-setter-keyword-block

GRAMMAR OF A PROTOCOL METHOD DECLARATION

protocol-method-declaration → function-head function-name generic-parameter-clause$_{opt}$ function-signature generic-where-clause$_{opt}$

GRAMMAR OF A PROTOCOL INITIALIZER DECLARATION

protocol-initializer-declaration → initializer-head generic-parameter-clause$_{opt}$ parameter-clause **throws**$_{opt}$ generic-where-clause$_{opt}$
protocol-initializer-declaration → initializer-head generic-parameter-clause$_{opt}$ parameter-clause **rethrows** generic-where-clause$_{opt}$

GRAMMAR OF A PROTOCOL SUBSCRIPT DECLARATION

protocol-subscript-declaration → subscript-head subscript-result generic-where-clause*opt* getter-setter-keyword-block

GRAMMAR OF A PROTOCOL ASSOCIATED TYPE DECLARATION

protocol-associated-type-declaration → attributes*opt* access-level-modifier*opt* **associatedtype** typealias-name type-inheritance-clause*opt* typealias-assignment*opt* generic-where-clause*opt*

GRAMMAR OF AN INITIALIZER DECLARATION

initializer-declaration → initializer-head generic-parameter-clause*opt* parameter-clause **throws***opt* generic-where-clause*opt* initializer-body

initializer-declaration → initializer-head generic-parameter-clause*opt* parameter-clause **rethrows** generic-where-clause*opt* initializer-body

initializer-head → attributes*opt* declaration-modifiers*opt* **init**

initializer-head → attributes*opt* declaration-modifiers*opt* **init ?**

initializer-head → attributes*opt* declaration-modifiers*opt* **init !**

initializer-body → code-block

GRAMMAR OF A DEINITIALIZER DECLARATION

deinitializer-declaration → attributes*opt* **deinit** code-block

GRAMMAR OF AN EXTENSION DECLARATION

extension-declaration → attributes*opt* access-level-modifier*opt* **extension** type-identifier type-inheritance-clause*opt* generic-where-clause*opt* extension-body

extension-body → **{** extension-members*opt* **}**

extension-members → extension-member extension-members*opt*

extension-member → declaration | compiler-control-statement

GRAMMAR OF A SUBSCRIPT DECLARATION

subscript-declaration → subscript-head subscript-result generic-where-clause*opt* code-block

subscript-declaration → subscript-head subscript-result generic-where-clause*opt* getter-setter-block

subscript-declaration → subscript-head subscript-result generic-where-clause*opt* getter-setter-keyword-block

subscript-head → attributes*opt* declaration-modifiers*opt* **subscript** generic-parameter-clause*opt* parameter-clause

subscript-result → **->** attributes$_{opt}$ type

GRAMMAR OF AN OPERATOR DECLARATION

operator-declaration → prefix-operator-declaration | postfix-operator-declaration | infix-operator-declaration

prefix-operator-declaration → **prefix operator** operator
postfix-operator-declaration → **postfix operator** operator
infix-operator-declaration → **infix operator** operator infix-operator-group$_{opt}$

infix-operator-group → **:** precedence-group-name

GRAMMAR OF A PRECEDENCE GROUP DECLARATION

precedence-group-declaration → **precedencegroup** precedence-group-name **{** precedence-group-attributes$_{opt}$ **}**

precedence-group-attributes → precedence-group-attribute precedence-group-attributes$_{opt}$
precedence-group-attribute → precedence-group-relation
precedence-group-attribute → precedence-group-assignment
precedence-group-attribute → precedence-group-associativity

precedence-group-relation → **higherThan :** precedence-group-names
precedence-group-relation → **lowerThan :** precedence-group-names

precedence-group-assignment → **assignment :** boolean-literal

precedence-group-associativity → **associativity : left**
precedence-group-associativity → **associativity : right**
precedence-group-associativity → **associativity : none**

precedence-group-names → precedence-group-name | precedence-group-name **,** precedence-group-names
precedence-group-name → identifier

GRAMMAR OF A DECLARATION MODIFIER

declaration-modifier → **class** | **convenience** | **dynamic** | **final** | **infix** | **lazy** | **optional** | **override** | **postfix** | **prefix** | **required** | **static** | **unowned** | **unowned (safe)** | **unowned (unsafe)** | **weak**
declaration-modifier → access-level-modifier
declaration-modifier → mutation-modifier
declaration-modifiers → declaration-modifier declaration-modifiers$_{opt}$

access-level-modifier → **private** | **private (set)**
access-level-modifier → **fileprivate** | **fileprivate (set)**
access-level-modifier → **internal** | **internal (set)**
access-level-modifier → **public** | **public (set)**
access-level-modifier → **open** | **open (set)**

mutation-modifier → **mutating** | **nonmutating**

Attributes

GRAMMAR OF AN ATTRIBUTE

attribute → **@** attribute-name attribute-argument-clause*opt*
attribute-name → identifier
attribute-argument-clause → **(** balanced-tokens*opt* **)**
attributes → attribute attributes*opt*

balanced-tokens → balanced-token balanced-tokens*opt*
balanced-token → **(** balanced-tokens*opt* **)**
balanced-token → **[** balanced-tokens*opt* **]**
balanced-token → **{** balanced-tokens*opt* **}**
balanced-token → Any identifier, keyword, literal, or operator
balanced-token → Any punctuation except **(**, **)**, **[**, **]**, **{**, or **}**

Patterns

GRAMMAR OF A PATTERN

pattern → wildcard-pattern type-annotation*opt*
pattern → identifier-pattern type-annotation*opt*
pattern → value-binding-pattern
pattern → tuple-pattern type-annotation*opt*
pattern → enum-case-pattern
pattern → optional-pattern
pattern → type-casting-pattern
pattern → expression-pattern

GRAMMAR OF A WILDCARD PATTERN

wildcard-pattern → **_**

GRAMMAR OF AN IDENTIFIER PATTERN

identifier-pattern → identifier

GRAMMAR OF A VALUE-BINDING PATTERN

value-binding-pattern → **var** pattern | **let** pattern

GRAMMAR OF A TUPLE PATTERN

tuple-pattern → **(** tuple-pattern-element-list$_{opt}$ **)**
tuple-pattern-element-list → tuple-pattern-element | tuple-pattern-element **,** tuple-pattern-element-list
tuple-pattern-element → pattern | identifier **:** pattern

GRAMMAR OF AN ENUMERATION CASE PATTERN

enum-case-pattern → type-identifier$_{opt}$ **.** enum-case-name tuple-pattern$_{opt}$

GRAMMAR OF AN OPTIONAL PATTERN

optional-pattern → identifier-pattern **?**

GRAMMAR OF A TYPE CASTING PATTERN

type-casting-pattern → is-pattern | as-pattern
is-pattern → **is** type
as-pattern → pattern **as** type

GRAMMAR OF AN EXPRESSION PATTERN

expression-pattern → expression

Generic Parameters and Arguments

GRAMMAR OF A GENERIC PARAMETER CLAUSE

generic-parameter-clause → **<** generic-parameter-list **>**

generic-parameter-list → generic-parameter | generic-parameter **,** generic-parameter-list
generic-parameter → type-name
generic-parameter → type-name **:** type-identifier
generic-parameter → type-name **:** protocol-composition-type

generic-where-clause → **where** requirement-list
requirement-list → requirement | requirement **,** requirement-list
requirement → conformance-requirement | same-type-requirement

conformance-requirement → type-identifier **:** type-identifier
conformance-requirement → type-identifier **:** protocol-composition-type
same-type-requirement → type-identifier **==** type

GRAMMAR OF A GENERIC ARGUMENT CLAUSE

generic-argument-clause → **<** generic-argument-list **>**
generic-argument-list → generic-argument | generic-argument **,** generic-argument-list
generic-argument → type

‹ Generic Parameters and Arguments Revision History ›