**Grammar of whitespace**

*whitespace* → *whitespace-item* *whitespace?*  
*whitespace-item* → *line-break*  
*whitespace-item* → *inline-space*  
*whitespace-item* → *comment*  
*whitespace-item* → *multiline-comment*  
*whitespace-item* → U+0000, U+000B, or U+000C

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

*inline-spaces* → *inline-space* *inline-spaces?*  
*inline-space* → U+0009 or U+0020

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

*comment-text* → *comment-text-item* *comment-text?*  
*comment-text-item* → Any Unicode scalar value except U+000A or U+000D

*multiline-comment-text* → *multiline-comment-text-item* *multiline-comment-text?*  
*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-characters?*  
*identifier* → **`** *identifier-head* *identifier-characters?* **`**  
*identifier* → *implicit-parameter-name*  
*identifier* → *property-wrapper-projection*  
*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?*

*implicit-parameter-name* → **$** *decimal-digits*  
*property-wrapper-projection* → **$** *identifier-characters*

**Grammar of a literal**

*literal* → *numeric-literal* | *string-literal* | *regular-expression-literal* | *boolean-literal* | *nil-literal*

*numeric-literal* → **-***?* *integer-literal* | **-***?* *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?*  
*binary-digit* → Digit 0 or 1  
*binary-literal-character* → *binary-digit* | **\_**  
*binary-literal-characters* → *binary-literal-character* *binary-literal-characters?*

*octal-literal* → **0o** *octal-digit* *octal-literal-characters?*  
*octal-digit* → Digit 0 through 7  
*octal-literal-character* → *octal-digit* | **\_**  
*octal-literal-characters* → *octal-literal-character* *octal-literal-characters?*

*decimal-literal* → *decimal-digit* *decimal-literal-characters?*  
*decimal-digit* → Digit 0 through 9  
*decimal-digits* → *decimal-digit* *decimal-digits?*  
*decimal-literal-character* → *decimal-digit* | **\_**  
*decimal-literal-characters* → *decimal-literal-character* *decimal-literal-characters?*

*hexadecimal-literal* → **0x** *hexadecimal-digit* *hexadecimal-literal-characters?*  
*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?*

**Grammar of a floating-point literal**

*floating-point-literal* → *decimal-literal* *decimal-fraction?* *decimal-exponent?*  
*floating-point-literal* → *hexadecimal-literal* *hexadecimal-fraction?* *hexadecimal-exponent*

*decimal-fraction* → **.** *decimal-literal*  
*decimal-exponent* → *floating-point-e* *sign?* *decimal-literal*

*hexadecimal-fraction* → **.** *hexadecimal-digit* *hexadecimal-literal-characters?*  
*hexadecimal-exponent* → *floating-point-p* *sign?* *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?* **"**  
*string-literal-closing-delimiter* → **"** *extended-string-literal-delimiter?*

*static-string-literal* → *string-literal-opening-delimiter* *quoted-text?* *string-literal-closing-delimiter*  
*static-string-literal* → *multiline-string-literal-opening-delimiter* *multiline-quoted-text?* *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?*

*quoted-text* → *quoted-text-item* *quoted-text?*  
*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?*  
*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?* *string-literal-closing-delimiter*  
*interpolated-string-literal* → *multiline-string-literal-opening-delimiter* *multiline-interpolated-text?* *multiline-string-literal-closing-delimiter*

*interpolated-text* → *interpolated-text-item* *interpolated-text?*  
*interpolated-text-item* → **\(** *expression* **)** | *quoted-text-item*

*multiline-interpolated-text* → *multiline-interpolated-text-item* *multiline-interpolated-text?*  
*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* *inline-spaces?* *line-break*

**Grammar of a regular expression literal**

*regular-expression-literal* → *regular-expression-literal-opening-delimiter* *regular-expression* *regular-expression-literal-closing-delimiter*  
*regular-expression* → Any regular expression

*regular-expression-literal-opening-delimiter* → *extended-regular-expression-literal-delimiter?* **/**  
*regular-expression-literal-closing-delimiter* → **/** *extended-regular-expression-literal-delimiter?*

*extended-regular-expression-literal-delimiter* → **#** *extended-regular-expression-literal-delimiter?*

**Grammar of operators**

*operator* → *operator-head* *operator-characters?*  
*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?*

*dot-operator-head* → **.**  
*dot-operator-character* → **.** | *operator-character*  
*dot-operator-characters* → *dot-operator-character* *dot-operator-characters?*

*infix-operator* → *operator*  
*prefix-operator* → *operator*  
*postfix-operator* → *operator*

[Types](https://docs.swift.org/swift-book/documentation/the-swift-programming-language/summaryofthegrammar#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* → *any-type*  
*type* → *self-type*  
*type* → **(** *type* **)**

**Grammar of a type annotation**

*type-annotation* → **:** *attributes?* **inout***?* *type*

**Grammar of a type identifier**

*type-identifier* → *type-name* *generic-argument-clause?* | *type-name* *generic-argument-clause?* **.** *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?* *function-type-argument-clause* **async***?* *throws-clause?* **->** *type*

*function-type-argument-clause* → **(** **)**  
*function-type-argument-clause* → **(** *function-type-argument-list* **...***?* **)**

*function-type-argument-list* → *function-type-argument* | *function-type-argument* **,** *function-type-argument-list*  
*function-type-argument* → *attributes?* **inout***?* *type* | *argument-label* *type-annotation*  
*argument-label* → *identifier*

*throws-clause* → **throws** | **throws** **(** *type* **)**

**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 boxed protocol type**

*boxed-protocol-type* → **any** *type*

**Grammar of a metatype type**

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

**Grammar of an Any type**

*any-type* → **Any**

**Grammar of a Self type**

*self-type* → **Self**

**Grammar of a type inheritance clause**

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

[Expressions](https://docs.swift.org/swift-book/documentation/the-swift-programming-language/summaryofthegrammar#Expressions)

**Grammar of an expression**

*expression* → *try-operator?* *await-operator?* *prefix-expression* *infix-expressions?* \

**Grammar of a prefix expression**

*prefix-expression* → *prefix-operator?* *postfix-expression*  
*prefix-expression* → *in-out-expression*

**Grammar of an in-out expression**

*in-out-expression* → **&** *primary-expression*

**Grammar of a try expression**

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

**Grammar of an await expression**

*await-operator* → **await**

**Grammar of an infix expression**

*infix-expression* → *infix-operator* *prefix-expression*  
*infix-expression* → *assignment-operator* *try-operator?* *await-operator?* *prefix-expression*  
*infix-expression* → *conditional-operator* *try-operator?* *await-operator?* *prefix-expression*  
*infix-expression* → *type-casting-operator*  
*infix-expressions* → *infix-expression* *infix-expressions?*

**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?*  
*primary-expression* → *literal-expression*  
*primary-expression* → *self-expression*  
*primary-expression* → *superclass-expression*  
*primary-expression* → *conditional-expression*  
*primary-expression* → *closure-expression*  
*primary-expression* → *parenthesized-expression*  
*primary-expression* → *tuple-expression*  
*primary-expression* → *implicit-member-expression*  
*primary-expression* → *wildcard-expression*  
*primary-expression* → *macro-expansion-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*

*array-literal* → **[** *array-literal-items?* **]**  
*array-literal-items* → *array-literal-item* **,***?* | *array-literal-item* **,** *array-literal-items*  
*array-literal-item* → *expression*

*dictionary-literal* → **[** *dictionary-literal-items* **]** | **[** **:** **]**  
*dictionary-literal-items* → *dictionary-literal-item* **,***?* | *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 conditional expression**

*conditional-expression* → *if-expression* | *switch-expression*

*if-expression* → **if** *condition-list* **{** *statement* **}** *if-expression-tail*  
*if-expression-tail* → **else** *if-expression*  
*if-expression-tail* → **else** **{** *statement* **}**

*switch-expression* → **switch** *expression* **{** *switch-expression-cases* **}**  
*switch-expression-cases* → *switch-expression-case* *switch-expression-cases?*  
*switch-expression-case* → *case-label* *statement*  
*switch-expression-case* → *default-label* *statement*

**Grammar of a closure expression**

*closure-expression* → **{** *attributes?* *closure-signature?* *statements?* **}**

*closure-signature* → *capture-list?* *closure-parameter-clause* **async***?* *throws-clause?* *function-result?* **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?*  
*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?* *identifier*  
*capture-list-item* → *capture-specifier?* *identifier* **=** *expression*  
*capture-list-item* → *capture-specifier?* *self-expression*  
*capture-specifier* → **weak** | **unowned** | **unowned(safe)** | **unowned(unsafe)**

**Grammar of an implicit member expression**

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

**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 macro-expansion expression**

*macro-expansion-expression* → **#** *identifier* *generic-argument-clause?* *function-call-argument-clause?* *trailing-closures?*

**Grammar of a key-path expression**

*key-path-expression* → **\** *type?* **.** *key-path-components*  
*key-path-components* → *key-path-component* | *key-path-component* **.** *key-path-components*  
*key-path-component* → *identifier* *key-path-postfixes?* | *key-path-postfixes*

*key-path-postfixes* → *key-path-postfix* *key-path-postfixes?*  
*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?* *trailing-closures*

*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-closures* → *closure-expression* *labeled-trailing-closures?*  
*labeled-trailing-closures* → *labeled-trailing-closure* *labeled-trailing-closures?*  
*labeled-trailing-closure* → *identifier* **:** *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?*  
*explicit-member-expression* → *postfix-expression* **.** *identifier* **(** *argument-names* **)**  
*explicit-member-expression* → *postfix-expression* *conditional-compilation-block*

*argument-names* → *argument-name* *argument-names?*  
*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](https://docs.swift.org/swift-book/documentation/the-swift-programming-language/summaryofthegrammar#Statements)

**Grammar of a statement**

*statement* → *expression* **;***?*  
*statement* → *declaration* **;***?*  
*statement* → *loop-statement* **;***?*  
*statement* → *branch-statement* **;***?*  
*statement* → *labeled-statement* **;***?*  
*statement* → *control-transfer-statement* **;***?*  
*statement* → *defer-statement* **;***?*  
*statement* → *do-statement* **;***?*  
*statement* → *compiler-control-statement*  
*statements* → *statement* *statements?*

**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***?* *pattern* **in** *expression* *where-clause?* *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?*  
*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?* **}**  
*switch-cases* → *switch-case* *switch-cases?*  
*switch-case* → *case-label* *statements*  
*switch-case* → *default-label* *statements*  
*switch-case* → *conditional-switch-case*

*case-label* → *attributes?* **case** *case-item-list* **:**  
*case-item-list* → *pattern* *where-clause?* | *pattern* *where-clause?* **,** *case-item-list*  
*default-label* → *attributes?* **default** **:**

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

*conditional-switch-case* → *switch-if-directive-clause* *switch-elseif-directive-clauses?* *switch-else-directive-clause?* *endif-directive*  
*switch-if-directive-clause* → *if-directive* *compilation-condition* *switch-cases?*  
*switch-elseif-directive-clauses* → *elseif-directive-clause* *switch-elseif-directive-clauses?*  
*switch-elseif-directive-clause* → *elseif-directive* *compilation-condition* *switch-cases?*  
*switch-else-directive-clause* → *else-directive* *switch-cases?*

**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?*

**Grammar of a continue statement**

*continue-statement* → **continue** *label-name?*

**Grammar of a fallthrough statement**

*fallthrough-statement* → **fallthrough**

**Grammar of a return statement**

*return-statement* → **return** *expression?*

**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** *throws-clause?* *code-block* *catch-clauses?*  
*catch-clauses* → *catch-clause* *catch-clauses?*  
*catch-clause* → **catch** *catch-pattern-list?* *code-block*  
*catch-pattern-list* → *catch-pattern* | *catch-pattern* **,** *catch-pattern-list*  
*catch-pattern* → *pattern* *where-clause?*

**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?* *else-directive-clause?* *endif-directive*

*if-directive-clause* → *if-directive* *compilation-condition* *statements?*  
*elseif-directive-clauses* → *elseif-directive-clause* *elseif-directive-clauses?*  
*elseif-directive-clause* → *elseif-directive* *compilation-condition* *statements?*  
*else-directive-clause* → *else-directive* *statements?*  
*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** **(** *import-path* **)**  
*platform-condition* → **targetEnvironment** **(** *environment* **)**

*operating-system* → **macOS** | **iOS** | **watchOS** | **tvOS** | **visionOS** | **Linux** | **Windows**  
*architecture* → **i386** | **x86\_64** | **arm** | **arm64**  
*swift-version* → *decimal-digits* *swift-version-continuation?*  
*swift-version-continuation* → **.** *decimal-digits* *swift-version-continuation?*  
*environment* → **simulator** | **macCatalyst**

**Grammar of a line control statement**

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

**Grammar of an availability condition**

*availability-condition* → **#available** **(** *availability-arguments* **)**  
*availability-condition* → **#unavailable** **(** *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* → **macCatalyst** | **macCatalystApplicationExtension**  
*platform-name* → **watchOS** | **watchOSApplicationExtension**  
*platform-name* → **tvOS** | **tvOSApplicationExtension**  
*platform-name* → **visionOS** | **visionOSApplicationExtension**  
*platform-version* → *decimal-digits*  
*platform-version* → *decimal-digits* **.** *decimal-digits*  
*platform-version* → *decimal-digits* **.** *decimal-digits* **.** *decimal-digits*

[Declarations](https://docs.swift.org/swift-book/documentation/the-swift-programming-language/summaryofthegrammar#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* → *actor-declaration*  
*declaration* → *protocol-declaration*  
*declaration* → *initializer-declaration*  
*declaration* → *deinitializer-declaration*  
*declaration* → *extension-declaration*  
*declaration* → *subscript-declaration*  
*declaration* → *operator-declaration*  
*declaration* → *precedence-group-declaration* \

**Grammar of a top-level declaration**

*top-level-declaration* → *statements?*

**Grammar of a code block**

*code-block* → **{** *statements?* **}**

**Grammar of an import declaration**

*import-declaration* → *attributes?* **import** *import-kind?* *import-path*

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

**Grammar of a constant declaration**

*constant-declaration* → *attributes?* *declaration-modifiers?* **let** *pattern-initializer-list*

*pattern-initializer-list* → *pattern-initializer* | *pattern-initializer* **,** *pattern-initializer-list*  
*pattern-initializer* → *pattern* *initializer?*  
*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?* *willSet-didSet-block*

*variable-declaration-head* → *attributes?* *declaration-modifiers?* **var**  
*variable-name* → *identifier*

*getter-setter-block* → *code-block*  
*getter-setter-block* → **{** *getter-clause* *setter-clause?* **}**  
*getter-setter-block* → **{** *setter-clause* *getter-clause* **}**  
*getter-clause* → *attributes?* *mutation-modifier?* **get** *code-block*  
*setter-clause* → *attributes?* *mutation-modifier?* **set** *setter-name?* *code-block*  
*setter-name* → **(** *identifier* **)**

*getter-setter-keyword-block* → **{** *getter-keyword-clause* *setter-keyword-clause?* **}**  
*getter-setter-keyword-block* → **{** *setter-keyword-clause* *getter-keyword-clause* **}**  
*getter-keyword-clause* → *attributes?* *mutation-modifier?* **get**  
*setter-keyword-clause* → *attributes?* *mutation-modifier?* **set**

*willSet-didSet-block* → **{** *willSet-clause* *didSet-clause?* **}**  
*willSet-didSet-block* → **{** *didSet-clause* *willSet-clause?* **}**  
*willSet-clause* → *attributes?* **willSet** *setter-name?* *code-block*  
*didSet-clause* → *attributes?* **didSet** *setter-name?* *code-block*

**Grammar of a type alias declaration**

*typealias-declaration* → *attributes?* *access-level-modifier?* **typealias** *typealias-name* *generic-parameter-clause?* *typealias-assignment*  
*typealias-name* → *identifier*  
*typealias-assignment* → **=** *type*

**Grammar of a function declaration**

*function-declaration* → *function-head* *function-name* *generic-parameter-clause?* *function-signature* *generic-where-clause?* *function-body?*

*function-head* → *attributes?* *declaration-modifiers?* **func**  
*function-name* → *identifier* | *operator*

*function-signature* → *parameter-clause* **async***?* *throws-clause?* *function-result?*  
*function-signature* → *parameter-clause* **async***?* **rethrows** *function-result?*  
*function-result* → **->** *attributes?* *type*  
*function-body* → *code-block*

*parameter-clause* → **(** **)** | **(** *parameter-list* **)**  
*parameter-list* → *parameter* | *parameter* **,** *parameter-list*  
*parameter* → *external-parameter-name?* *local-parameter-name* *parameter-type-annotation* *default-argument-clause?*  
*parameter* → *external-parameter-name?* *local-parameter-name* *parameter-type-annotation*  
*parameter* → *external-parameter-name?* *local-parameter-name* *parameter-type-annotation* **...**

*external-parameter-name* → *identifier*  
*local-parameter-name* → *identifier*  
*parameter-type-annotation* → **:** *attributes?* *parameter-modifier?* *type*  
*parameter-modifier* → **inout** | **borrowing** | **consuming** *default-argument-clause* → **=** *expression*

**Grammar of an enumeration declaration**

*enum-declaration* → *attributes?* *access-level-modifier?* *union-style-enum*  
*enum-declaration* → *attributes?* *access-level-modifier?* *raw-value-style-enum*

*union-style-enum* → **indirect***?* **enum** *enum-name* *generic-parameter-clause?* *type-inheritance-clause?* *generic-where-clause?* **{** *union-style-enum-members?* **}**  
*union-style-enum-members* → *union-style-enum-member* *union-style-enum-members?*  
*union-style-enum-member* → *declaration* | *union-style-enum-case-clause* | *compiler-control-statement*  
*union-style-enum-case-clause* → *attributes?* **indirect***?* **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?*  
*enum-name* → *identifier*  
*enum-case-name* → *identifier*

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

**Grammar of a structure declaration**

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

*struct-members* → *struct-member* *struct-members?*  
*struct-member* → *declaration* | *compiler-control-statement*

**Grammar of a class declaration**

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

*class-members* → *class-member* *class-members?*  
*class-member* → *declaration* | *compiler-control-statement*

**Grammar of an actor declaration**

*actor-declaration* → *attributes?* *access-level-modifier?* **actor** *actor-name* *generic-parameter-clause?* *type-inheritance-clause?* *generic-where-clause?* *actor-body*  
*actor-name* → *identifier*  
*actor-body* → **{** *actor-members?* **}**

*actor-members* → *actor-member* *actor-members?*  
*actor-member* → *declaration* | *compiler-control-statement*

**Grammar of a protocol declaration**

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

*protocol-members* → *protocol-member* *protocol-members?*  
*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?* *function-signature* *generic-where-clause?*

**Grammar of a protocol initializer declaration**

*protocol-initializer-declaration* → *initializer-head* *generic-parameter-clause?* *parameter-clause* *throws-clause?* *generic-where-clause?*  
*protocol-initializer-declaration* → *initializer-head* *generic-parameter-clause?* *parameter-clause* **rethrows** *generic-where-clause?*

**Grammar of a protocol subscript declaration**

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

**Grammar of a protocol associated type declaration**

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

**Grammar of an initializer declaration**

*initializer-declaration* → *initializer-head* *generic-parameter-clause?* *parameter-clause* **async***?* *throws-clause?* *generic-where-clause?* *initializer-body*  
*initializer-declaration* → *initializer-head* *generic-parameter-clause?* *parameter-clause* **async***?* **rethrows** *generic-where-clause?* *initializer-body*  
*initializer-head* → *attributes?* *declaration-modifiers?* **init**  
*initializer-head* → *attributes?* *declaration-modifiers?* **init** **?**  
*initializer-head* → *attributes?* *declaration-modifiers?* **init** **!**  
*initializer-body* → *code-block*

**Grammar of a deinitializer declaration**

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

**Grammar of an extension declaration**

*extension-declaration* → *attributes?* *access-level-modifier?* **extension** *type-identifier* *type-inheritance-clause?* *generic-where-clause?* *extension-body*  
*extension-body* → **{** *extension-members?* **}**

*extension-members* → *extension-member* *extension-members?*  
*extension-member* → *declaration* | *compiler-control-statement*

**Grammar of a subscript declaration**

*subscript-declaration* → *subscript-head* *subscript-result* *generic-where-clause?* *code-block*  
*subscript-declaration* → *subscript-head* *subscript-result* *generic-where-clause?* *getter-setter-block*  
*subscript-declaration* → *subscript-head* *subscript-result* *generic-where-clause?* *getter-setter-keyword-block*  
*subscript-head* → *attributes?* *declaration-modifiers?* **subscript** *generic-parameter-clause?* *parameter-clause*  
*subscript-result* → **->** *attributes?* *type*

**Grammar of a macro declaration**

*macro-declaration* → *macro-head* *identifier* *generic-parameter-clause?* *macro-signature* *macro-definition?* *generic-where-clause*  
*macro-head* → *attributes?* *declaration-modifiers?* **macro**  
*macro-signature* → *parameter-clause* *macro-function-signature-result?*  
*macro-function-signature-result* → **->** *type*  
*macro-definition* → **=** *expression*

**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?*

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

**Grammar of a precedence group declaration**

*precedence-group-declaration* → **precedencegroup** *precedence-group-name* **{** *precedence-group-attributes?* **}**

*precedence-group-attributes* → *precedence-group-attribute* *precedence-group-attributes?*  
*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-modifier* → *actor-isolation-modifier*  
*declaration-modifiers* → *declaration-modifier* *declaration-modifiers?*

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

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

*actor-isolation-modifier* → **nonisolated**

[Attributes](https://docs.swift.org/swift-book/documentation/the-swift-programming-language/summaryofthegrammar#Attributes)

**Grammar of an attribute**

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

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

[Patterns](https://docs.swift.org/swift-book/documentation/the-swift-programming-language/summaryofthegrammar#Patterns)

**Grammar of a pattern**

*pattern* → *wildcard-pattern* *type-annotation?*  
*pattern* → *identifier-pattern* *type-annotation?*  
*pattern* → *value-binding-pattern*  
*pattern* → *tuple-pattern* *type-annotation?*  
*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?* **)**  
*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?* **.** *enum-case-name* *tuple-pattern?*

**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](https://docs.swift.org/swift-book/documentation/the-swift-programming-language/summaryofthegrammar#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*