翻译：StanZhai 校对：xielingwang

语法总结

本页包含内容：

语句

语句语法 *statement* → *expression* **;** *opt* *statement* → *declaration* **;** *opt* *statement* → *loop-statement* **;** *opt* *statement* → *branch-statement* **;** *opt* *statement* → *labeled-statement* *statement* → *control-transfer-statement* **;** *opt* *statements* → *statement* *statements* *opt*

循环语句语法 *loop-statement* → *for-statement* *loop-statement* → *for-in-statement* *loop-statement* → *while-statement* *loop-statement* → *do-while-statement*

For 循环语法 *for-statement* → **for** *for-init* *opt* **;** *expression* *opt* **;** *expression* *opt* *code-block* *for-statement* → **for** **(** *for-init* *opt* **;** *expression* *opt* **;** *expression* *opt* **)** *code-block* *for-init* → *variable-declaration* | *expression-list*

For-In 循环语法 *for-in-statement* → **for** *pattern* **in** *expression code-block*

While 循环语法 *while-statement* → **while** *while-condition code-block while-condition* → *expression* | *declaration*

Do-While 循环语法 *do-while-statement* → **do** *code-block* **while** *while-condition*

分支语句语法 *branch-statement* → *if-statement branch-statement* → *switch-statement*

If 语句语法 *if-statement* → **if** *if-condition code-block else-clause opt if-condition* → *expression* | *declaration else-clause* → **else** *code-block* | **else** *if-statement*

Switch 语句语法 *switch-statement* → **switch** *expression* **{** *switch-cases opt* **}** *switch-cases* → *switch-case switch-cases opt switch-case* → *case-label statements* | *default-label statements switch-case* → *case-label* **;** | *default-label* **;** *case-label* → **case** *case-item-list* **:** *case-item-list* → *pattern guard-clause opt* | *pattern guard-clause opt* **,** *case-item-list default-label* → **default** **:** *guard-clause* → **where** *guard-expression guard-expression* → *expression*

带标签的语句语法 *labeled-statement* → *statement-label loop-statement* | *statement-label switch-statement statement-label* → *label-name* **:** *label-name* → *identifier*

控制传递语句语法 *control-transfer-statement* → *break-statement control-transfer-statement* → *continue-statement control-transfer-statement* → *fallthrough-statement control-transfer-statement* → *return-statement*

Break 语句语法 *break-statement* → **break** *label-name opt*

Continue 语句语法 *continue-statement* → **continue** *label-name opt*

Fallthrough 语句语法 *fallthrough-statement* → **fallthrough**

Return 语句语法 *return-statement* → **return** *expression opt*

泛型参数

泛型形参子句语法 *generic-parameter-clause* → **<** *generic-parameter-list requirement-clause opt* **>** *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 requirement-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-identifier*

泛型实参子句语法 *generic-argument-clause* → **<** *generic-argument-list* **>** *generic-argument-list* → *generic-argument* | *generic-argument* **,** *generic-argument-list generic-argument* → *type*

声明 (Declarations)

声明语法 *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* *declarations* → *declaration declarations opt* *declaration-specifiers* → *declaration-specifier declaration-specifiers opt* *declaration-specifier* → **class** | **mutating** | **nonmutating** | **override** | **static** | **unowned** | **unowned(safe)** | **unowned(unsafe)** | **weak**

顶级声明语法 *top-level-declaration* → *statements opt*

代码块语法 *code-block* → { *statements opt* }

Import 声明语法 *import-declaration* → *attributes opt* **import** *import-kind opt import-path* *import-kind* → **typealias** | **struct** | **class** | **enum** | **protocol** | **var** | **func** *import-path* → *import-path-identifier* | *import-path-identifier* **.** *import-path* *import-path-identifier* → *identifier* | *operator*

常数声明语法 *constant-declaration* → *attributes opt declaration-specifiers opt* **let** *pattern-initializer-list* *pattern-initializer-list* → *pattern-initializer* | *pattern-initializer* **,** *pattern-initializer-list* *pattern-initializer* → *pattern initializer opt* *initializer* → **=** *expression*

变量声明语法 *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 type-annotation initializer opt willSet-didSet-block* *variable-declaration-head* → *attributes opt declaration-specifiers opt* **var** *variable-name* → *identifier* *getter-setter-block* → { *getter-clause setter-clause opt* } *getter-setter-block* → { *setter-clause getter-clause* } *getter-clause* → *attributes opt* **get** *code-block* *setter-clause* → *attributes 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* **get** *setter-*

keyword-clause → *attributes* *opt* **set** *willSet-didSet-block* → { *willSet-clause* *didSet-clause* *opt* } *willSet-didSet-block* → { *didSet-clause* *willSet-clause* } *willSet-clause* → *attributes* *opt* **willSet** *setter-name* *opt* *code-block* *didSet-clause* → *attributes* *opt* **didSet** *setter-name* *opt* *code-block*

类型别名声明语法 *typealias-declaration* → *typealias-head* *typealias-assignment* *typealias-head* → **typealias** *typealias-name* *typealias-name* → *identifier* *typealias-assignment* → **=** *type*

函数声明语法 *function-declaration* → *function-head* *function-name* *generic-parameter-clause* *opt* *function-signature* *function-body* *function-head* → *attributes* *opt* *declaration-specifiers* *opt* **func** *function-name* → *identifier* | *operator* *function-signature* → *parameter-clauses* *function-result* *opt* *function-result* → **->** *attributes* *opt* *type* *function-body* → *code-block* *parameter-clauses* → *parameter-clause* *parameter-clauses* *opt* *parameter-clause* → **( )** | **(** *parameter-list* **...** *opt* **)** *parameter-list* → *parameter* | *parameter* **,** *parameter-list* *parameter* → **inout** *opt* **let** *opt* **#** *opt* *parameter-name* *local-parameter-name* *opt* *type-annotation* *default-argument-clause* *opt* *parameter* → **inout** *opt* **var** **#** *opt* *parameter-name* *local-parameter-name* *opt* *type-annotation* *default-argument-clause* *opt* *parameter* → *attributes* *opt* *type* *parameter-name* → *identifier* | **_** *local-parameter-name* → *identifier* | **_** *default-argument-clause* → **=** *expression*

枚举声明语法 *enum-declaration* → *attributes* *opt* *union-style-enum* | *attributes* *opt* *raw-value-style-enum* *union-style-enum* → *enum-name* *generic-parameter-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* *union-style-enum-case-clause* → *attributes* *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-name* *generic-parameter-clause* *opt* **:** *type-identifier* { *raw-value-style-enum-members* *opt* } *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* *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* → **=** *literal*

结构体声明语法 *struct-declaration* → *attributes opt* **struct** *struct-name generic-parameter-clause opt type-inheritance-clause opt struct-body struct-name* → *identifier struct-body* → { *declarations opt* }

类声明语法 *class-declaration* → *attributes opt* **class** *class-name generic-parameter-clause opt type-inheritance-clause opt class-body class-name* → *identifier class-body* → { *declarations opt* }

协议声明语法 *protocol-declaration* → *attributes opt* **protocol** *protocol-name type-inheritance-clause opt protocol-body protocol-name* → *identifier protocol-body* → { *protocol-member-declarations opt* } *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-declarations* → *protocol-member-declaration protocol-member-declarations opt*

协议属性声明语法 *protocol-property-declaration* → *variable-declaration-head variable-name type-annotation getter-setter-keyword-block*

协议方法声明语法 *protocol-method-declaration* → *function-head function-name generic-parameter-clause opt function-signature*

协议构造函数声明语法 *protocol-initializer-declaration* → *initializer-head generic-parameter-clause opt parameter-clause*

协议附属脚本声明语法 *protocol-subscript-declaration* → *subscript-head* *subscript-result getter-setter-keyword-block*

协议关联类型声明语法 *protocol-associated-type-declaration* → *typealias-head type-inheritance-clause opt typealias-assignment opt*

构造函数声明语法 *initializer-declaration* → *initializer-head generic-parameter-clause opt parameter-clause initializer-body initializer-head* → *attributes opt* **convenience** *opt* **init** *initializer-body* → *code-block*

析构函数声明语法 *deinitializer-declaration* → *attributes opt* **deinit** *code-block*

扩展声明语法 *extension-declaration* → **extension** *type-identifier type-inheritance-clause opt extension-body extension-body* → { *declarations opt* }

附属脚本声明语法 *subscript-declaration* → *subscript-head subscript-result code-block subscript-declaration* → *subscript-head subscript-result getter-setter-block subscript-declaration* → *subscript-head subscript-result getter-setter-keyword-block subscript-head* → *attributes opt* **subscript** *parameter-clause subscript-result* → **->** *attributes opt type*

运算符声明语法 *operator-declaration* → *prefix-operator-declaration* | *postfix-operator-declaration* | *infix-operator-declaration prefix-operator-declaration* → **operator prefix** *operator* { } *postfix-operator-declaration* → **operator postfix** *operator* { } *infix-operator-declaration* → **operator infix** *operator* { *infix-operator-attributes opt* } *infix-operator-attributes* → *precedence-clause opt associativity-clause opt precedence-clause* → **precedence** *precedence-level precedence-level* → Digit 0 through 255 *associativity-clause* → **associativity** *associativity associativity* → **left** | **right** | **none**

模式

模式语法 *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* → *type-casting-pattern pattern* → *expression-pattern*

通配符模式语法 *wildcard-pattern* → _

标识符模式语法 *identifier-pattern* → *identifier*

值绑定模式语法 *value-binding-pattern* → **var** *pattern* | **let** *pattern*

元组模式语法 *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*

枚举用例模式语法 *enum-case-pattern* → *type-identifier opt* **.** *enum-case-name tuple-pattern opt*

类型转换模式语法 *type-casting-pattern* → *is-pattern* | *as-pattern is-pattern* → **is** *type as-pattern* → *pattern* **as** *type*

表达式模式语法 *expression-pattern* → *expression*

特性

特性语法 *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 }

表达式

表达式语法 *expression* → *prefix-expression binary-expressions opt expression-list* → *expression* | *expression* **,** *expression-list*

前缀表达式语法 *prefix-expression* → *prefix-operator opt postfix-expression prefix-expression* → *in-out-expression in-out-expression* → **&** *identifier*

二进制表达式语法 *binary-expression* → *binary-operator prefix-expression binary-expression* → *assignment-operator prefix-expression binary-expression* → *conditional-operator prefix-expression binary-expression* → *type-casting-operator binary-expressions* → *binary-expression binary-expressions opt*

赋值运算符语法 *assignment-operator* → **=**

条件运算符语法 *conditional-operator* → **?** *expression* **:**

类型转换运算符语法 *type-casting-operator* → **is** *type* | **as** **?** *opt type*

主表达式语法 *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* → *implicit-member-expression primary-expression* → *wildcard-expression*

字面量表达式语法 *literal-expression* → *literal literal-expression* → *array-literal* | *dictionary-literal literal-expression* → **__FILE__** | **__LINE__** | **__COLUMN__** | **__FUNCTION__** *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*

Self 表达式语法 *self-expression* → **self** *self-expression* → **self . ** *identifier* *self-expression* → **self [** *expression* **]** *self-expression* → **self . init**

超类表达式语法 *superclass-expression* → *superclass-method-expression* | *superclass-subscript-expression* | *superclass-initializer-expression* *superclass-method-expression* → **super .** *identifier* *superclass-subscript-expression* → **super [** *expression* **]** *superclass-initializer-expression* → **super . init**

闭包表达式语法 *closure-expression* → **{** *closure-signature* *opt* *statements* **}** *closure-signature* → *parameter-clause* *function-result* *opt* **in** *closure-signature* → *identifier-list* *function-result* *opt* **in** *closure-signature* → *capture-list* *parameter-clause* *function-result* *opt* **in** *closure-signature* → *capture-list* *identifier-list* *function-result* *opt* **in** *closure-signature* → *capture-list* **in** *capture-list* → **[** *capture-specifier* *expression* **]** *capture-specifier* → **weak** | **unowned** | **unowned(safe)** | **unowned(unsafe)**

隐式成员表达式语法 *implicit-member-expression* → **.** *identifier*

带圆括号的表达式语法 *parenthesized-expression* → **(** *expression-element-list* *opt* **)** *expression-element-list* → *expression-element* | *expression-element* **,** *expression-element-list* *expression-element* → *expression* | *identifier* **:** *expression*

通配符表达式语法 *wildcard-expression* → **_**

后缀表达式语法 *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 → *dynamic-type-expression* *postfix-expression* → *subscript-expression* *postfix-expression* → *forced-value-expression* *postfix-expression* → *optional-chaining-expression*

函数调用表达式语法 *function-call-expression* → *postfix-expression* *parenthesized-expression* *function-call-expression* → *postfix-expression* *parenthesized-expression* *opt* *trailing-closure* *trailing-closure* → *closure-expression*

初始化表达式语法 *initializer-expression* → *postfix-expression* **. init**

显式成员表达式语法 *explicit-member-expression* → *postfix-expression* **.** *decimal-digit* *explicit-member-expression* → *postfix-expression* **.** *identifier* *generic-argument-clause* *opt*

Self 表达式语法 *postfix-self-expression* → *postfix-expression* **. self**

动态类型表达式语法 *dynamic-type-expression* → *postfix-expression* **. dynamicType**

附属脚本表达式语法 *subscript-expression* → *postfix-expression* **[** *expression-list* **]**

Forced-value 表达式语法 *forced-value-expression* → *postfix-expression* **!**

可选链表达式语法 *optional-chaining-expression* → *postfix-expression* **?**

词法结构

标识符语法 *identifier* → *identifier-head* *identifier-characters* *opt* *identifier* → `

identifier-head identifier-characters opt ` *identifier* → *implicit-parameter-name identifier-list* → *identifier* | *identifier* **,** *identifier-list identifier-head* → Upper- or lowercase letter A through Z *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*

字面量语法 *literal* → *integer-literal* | *floating-point-literal* | *string-literal*

整形字面量语法 *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*

浮点型字面量语法 *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-literal opt hexadecimal-exponent* → *floating-point-p sign opt hexadecimal-literal floating-point-e* → **e** | **E** *floating-point-p* → **p** | **P** *sign* → **+** | **-**

字符型字面量语法 *string-literal* → **"** *quoted-text* **"** *quoted-text* → *quoted-text-item quoted-text opt quoted-text-item* → *escaped-character quoted-text-item* → **(** *expression* **)** *quoted-text-item* → Any Unicode extended grapheme cluster except ", \, U+000A, or U+000D *escaped-character* → **\0** | \ | **\t** | **\n** | **\r** | **\"** | **\'** *escaped-character* → **\x** *hexadecimal-digit hexadecimal-digit escaped-character* → **\u** *hexadecimal-digit hexadecimal-digit hexadecimal-digit hexadecimal-digit escaped-character* → **\U** *hexadecimal-digit hexadecimal-digit hexadecimal-digit hexadecimal-digit hexadecimal-digit hexadecimal-digit hexadecimal-digit hexadecimal-digit*

运算符语法语法 *operator* → *operator-character operator opt operator-character* → **/** | **=** | **-** | **+** | **!** | ***** | **%** | **<** | **>** | **&** | **|** | **^** | **~** | **.** *binary-operator* → *operator prefix-operator* → *operator postfix-operator* → *operator*

类型

类型语法 *type* → *array-type* | *function-type* | *type-identifier* | *tuple-type* | *optional-type* | *implicitly-unwrapped-optional-type* | *protocol-composition-type* | *metatype-type*

类型标注语法 *type-annotation* → **:** *attributes opt type*

类型标识语法 *type-identifier* → *type-name generic-argument-clause opt* | *type-name generic-argument-clause opt* **.** *type-identifier type-name* → *identifier*

元组类型语法 *tuple-type* → **(** *tuple-type-body opt* **)** *tuple-type-body* → *tuple-type-element-list* **...** *opt tuple-type-element-list* → *tuple-type-element* | *tuple-type-element* **,** *tuple-type-element-list tuple-type-element* → *attributes opt* **inout** *opt type* | **inout** *opt element-name type-annotation element-name* → *identifier*

函数类型语法 *function-type* → *type* **->** *type*

数组类型语法 *array-type* → *type* **[ ]** | *array-type* **[ ]**

可选类型语法 *optional-type* → *type* **?**

隐式解析可选类型语法 *implicitly-unwrapped-optional-type* → *type* **!**

协议合成类型语法 *protocol-composition-type* → **protocol <** *protocol-identifier-list opt* **>** *protocol-identifier-list* → *protocol-identifier* | *protocol-identifier* **,** *protocol-identifier-list protocol-identifier* → *type-identifier*

元类型语法 *metatype-type* → *type* **. Type** | *type* **. Protocol**

类型继承子句语法 *type-inheritance-clause* → **:** *type-inheritance-list type-inheritance-list* → *type-identifier* | *type-identifier* **,** *type-inheritance-list*