

**Elm**  
Application  
Package

**Application**  
ElmJson ElmFile

**Package**  
ElmJson ElmFile

**ElmFile**  
Module

**Module**  
Definition Import Declaration

**Definition**  
module ModuleName exposing ( .. ExposedItem )  
port

**Import**  
import ModuleName  
as ModuleAliasName exposing ( .. ExposedItem )

**ExposedItem**  
TypeName  
(..)

ValueName  
(OperatorSymbol)

**Declaration**  
type TypeName  
TypeVariableName = TypeVariantName  
SingleTypeExpression

type alias TypeName  
TypeVariableName = TypeExpression

ValueName : TypeExpression  
ValueName DestructuringPattern = Expression  
port ValueName : TypeExpression

**TypeExpression**

TypeName  
ModuleName.TypeName SingleTypeExpression -> TypeExpression  
SingleTypeExpression

**SingleTypeExpression**

TypeName  
ModuleName.  
TypeVariableName  
ConstrainedType  
()  
( TypeExpression, TypeExpression )  
( TypeExpression, TypeExpression, TypeExpression )  
{  
TypeVariableName | FieldName : TypeExpression  
(TypeExpression)

Elm

**ModuleName**  
UppercaseName

**TypeName**  
UppercaseName

**TypeVariantName**  
UppercaseName

**TypeVariableName**  
LowercaseName

**ConstrainedType**  
number  
appendable  
comparable  
compappend

**ValueName**  
LowercaseName

**FieldName**  
LowercaseName

**UppercaseName**  
[A-Z][0-9A-Za-z\_]\*

**LowercaseName**  
[a-z][0-9A-Za-z\_]\*

Name

**Expression**

let ValueName DestructingPattern = Expression in Expression  
DestructingPattern ValueName : TypeExpression  
if Expression then Expression else Expression  
case Expression of Pattern -> Expression  
\ DestructingPattern -> Expression  
SingleExpression  
OperatorSymbol

**SingleExpression**

Number  
'Char'  
"String"  
[]  
[ Expression ]  
()  
( Expression, Expression )  
( Expression, Expression, Expression )  
{  
FieldName = Expression  
ValueName | FieldName = Expression  
[glsl| OpenGLShadingLanguage |]  
TypeVariantName  
ModuleName.  
ValueName FieldAccessor  
ModuleName.  
.FieldName  
(OperatorSymbol)  
-(Expression)  
(Expression) FieldAccessor

**FieldAccessor**

.FieldName

**Pattern**

SinglePattern  
as ValueName  
::

**SinglePattern**

ValueName  
-  
Number  
'Char'  
"String"  
[]  
[ Pattern ]  
()  
( Pattern, Pattern )  
( Pattern, Pattern, Pattern )  
{  
FieldName  
TypeVariantName  
SinglePattern  
ModuleName.  
(Pattern)

**DestructingPattern**

ValueName  
-  
()  
( DestructingPattern, DestructingPattern )  
( DestructingPattern, DestructingPattern, DestructingPattern )  
{  
FieldName  
TypeVariantName  
ModuleName.  
TypeVariantName  
DestructingPattern  
( DestructingPattern as ValueName )  
(DestructingPattern)

Value