Expression Grammar

Expr::= ConditionalExpr | LogicalOrExpr
ConditionalExpr ::= **?** Expr **->** Expr **,** Expr
LogicalOrExpr ::= LogicalAndExpr ((**|** | **||**) LogicalAndExpr)*
LogicalAndExpr ::= ComparisonExpr ((**&** | **&&**) ComparisonExpr)*
ComparisonExpr ::= PowExpr ((**<** | **>** | **==** | **<=** | **>=**) PowExpr)*
PowExpr ::= AdditiveExpr ****** PowExpr | AdditiveExpr
AdditiveExpr ::= MultiplicativeExpr ((**+** | **-**) MultiplicativeExpr)*
MultiplicativeExpr ::= UnaryExpr ((***** | **/** | **%**) UnaryExpr)*
UnaryExpr ::= (**!** | **-** | **width** | **height**) UnaryExpr | PostfixExpr
PostfixExpr::= PrimaryExpr (PixelSelector | ε) (ChannelSelector | ε)
PrimaryExpr ::=**STRING_LIT** | **NUM_LIT** | **IDENT** | **(** Expr **)** | **CONST** |
ExpandedPixelExpr
ChannelSelector ::= **: red** | **: green** | **: blue**
PixelSelector ::= **[** Expr **,** Expr **]**
ExpandedPixelExpr ::= **[** Expr **,** Expr **,** Expr **]**

The above grammar is a subset of the grammar for the programming language. The symbols in red correspond to token Kinds. In some cases (STRING_LIT, NUM_LIT, etc) the grammar uses the actual name of the constant in the Kind enum. In other cases, the text of the symbol itself is used for brevity. For example,
PixelSelector ::= **[** Expr **,** Expr **]**
rather than
PixelSelector ::= **LSQUARE** Expr **COMMMA** Expr **RSQUARE**