RegexExpression

```
syntax RegexExpression
{
  | Choice              = LeftRule=RegexExpression ^ 10 RightRules=("|" RegexExpression ^ 10)+
  | Sequence             = LeftRule=RegexExpression ^ 20 RightRules=(RegexExpression ^ 20)+
  | Optional            = RegexExpression "?" precedence 30
  | ZeroOrMany            = RegexExpression "*" precedence 30
  | OneOrMany             = RegexExpression "+" precedence 30
  | Invert              = "~" RegexExpression precedence 25
  | Subtract            = Rule1=RegexExpression "-" Rule2=RegexExpression precedence 22
  | Char                = CharLiteral
  | String              = StringLiteral
  | Call                = QualifiedName
  | Rounds               = "(" RegexExpression ")"
  | Range               = "[" (Range; ",")+ "]"
  | InvertedRange         = "[" "^" (Range; ",")+ "]"
  | ZeroOrManyWithSeparator = "(" RegexExpression ";" Separator ")" "*"
  | OneOrManyWithSeparator  = "(" RegexExpression ";" Separator ")" "+"
}
```

`RegexExpression` describes the `RegexRule` body, and is also used in some places where a regular expression should be specified.

The result of the `RegexExpression` parsing is the `NSpan` type that describes a text span. `AST nodes` are not created for `RegexExpression`.

See also

RegexRule
Choice
Sequence
Optional
ZeroOrMany
OneOrMany
Invert
Subtract
Char
String
Call
Rounds
Range
InvertedRange
ZeroOrManyWithSeparator
OneOrManyWithSeparator
NSpan