JavaScript is disabled on your browser.

- **Prev Class**
- **Next Class**

- Frames
- No Frames

- All Classes

- Summary:
- Nested |
- Field |
- Constr |
- Method

- Detail:
- Field |
- Constr |
- Method

jminusminus

Interface JLhs

- All Known Implementing Classes:
  JArrayExpression, JFieldSelection, JVariable

```
interface JLhs
```

The type of any expression that can appear on the lhs of an assignment statement, i.e., JVariable, JFieldSelection, JArrayExpression.

- ## Method Summary

Methods

Modifier and Type	Method and Description
`JExpression`	`analyzeLhs(Context context)` Analyze the lhs of an assignment.
`void`	`codegenDuplicateRvalue(CLEmitter output)` Generate the code required for duplicating the Rvalue that is on the stack becuase it is to be used in a surrounding expression, as in a[i] = x = or x = y--.
`void`	`codegenLoadLhsLvalue(CLEmitter output)` The up front code necessary for implementing an assignment; it generates code to load onto the stack any part of the lhs variable that must be there.
`void`	`codegenLoadLhsRvalue(CLEmitter output)` Generate the code required for loading an Rvalue for this variable, as in a +=.
`void`	`codegenStore(CLEmitter output)`

Generate the code required for doing the actual assignment.

- ## Method Detail

  - ### analyzeLhs

`JExpressionanalyzeLhs(Contextcontext)`

Analyze the lhs of an assignment. This is very much like analyze() but perhaps a little more selective here and there.

Parameters:

`context` - context in which names are resolved.

Returns:

the analyzed (and possibly rewritten) AST subtree.

  - ### codegenLoadLhsLvalue

`voidcodegenLoadLhsLvalue(CLEmitteroutput)`

The up front code necessary for implementing an assignment; it generates code to load onto the stack any part of the lhs variable that must be there. For example, in a[i] = x, code must be generated to load the array (a) and the index (i).

Parameters:

`output` - the code emitter (basically an abstraction for producing the .class file).

  - ### codegenLoadLhsRvalue

`voidcodegenLoadLhsRvalue(CLEmitteroutput)`

Generate the code required for loading an Rvalue for this variable, as in a +=.

Parameters:

`output` - the code emitter (basically an abstraction for producing the .class file).

  - ### codegenDuplicateRvalue

`voidcodegenDuplicateRvalue(CLEmitteroutput)`

Generate the code required for duplicating the Rvalue that is on the stack becuase it is to be used in a surrounding expression, as in a[i] = x = or x = y--.

Parameters:

`output` - the code emitter (basically an abstraction for producing the .class file).

  - ### codegenStore

`voidcodegenStore(CLEmitteroutput)`

Generate the code required for doing the actual assignment.

Parameters:

`output` - the code emitter (basically an abstraction for producing the .class file).

- **Prev Class**
- **Next Class**

- Frames
- No Frames

- All Classes

- Summary:
- Nested |
- Field |
- Constr |
- Method

- Detail:
- Field |
- Constr |
- Method