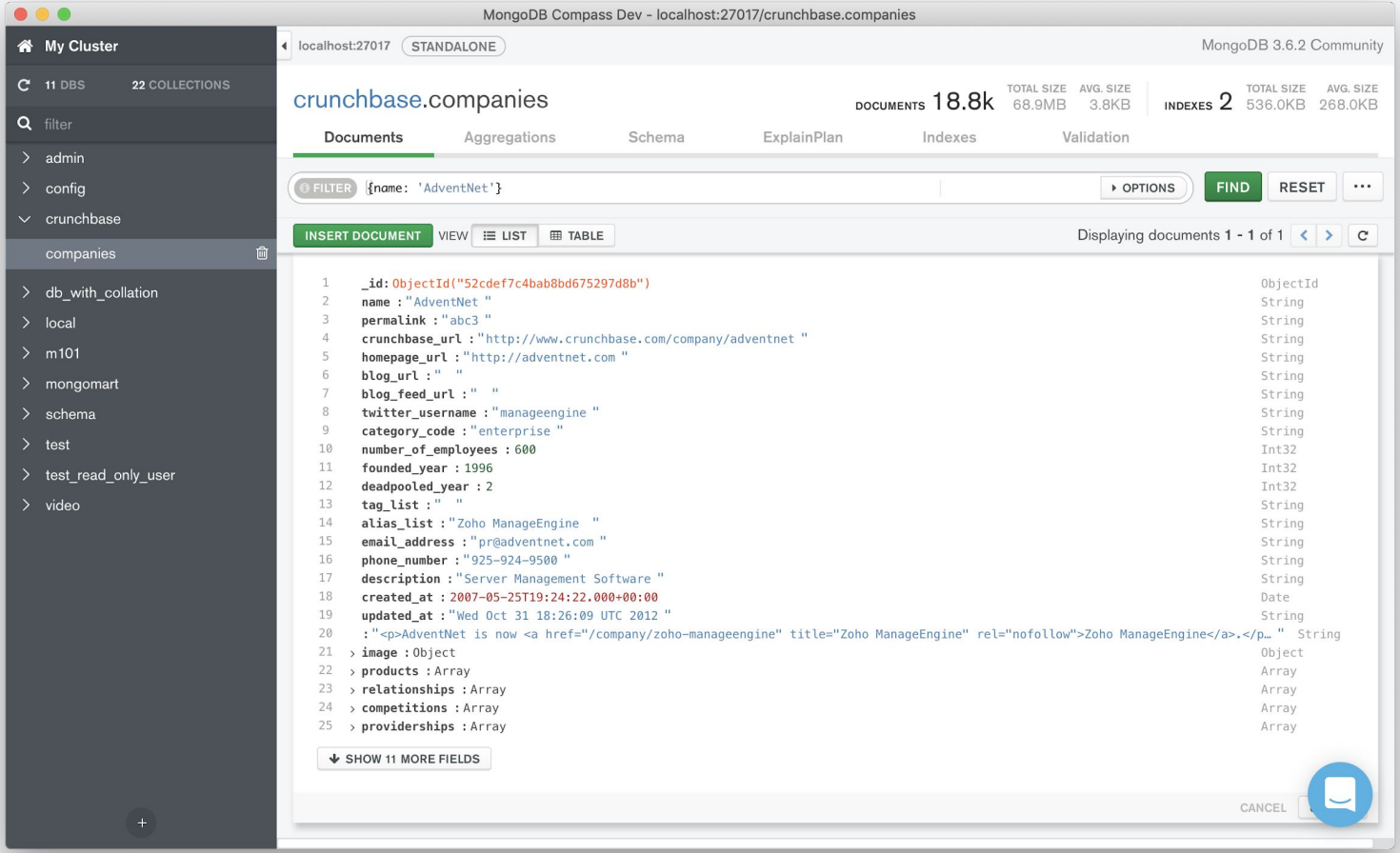
COMPILER

IN JAVASCRIPT USING ANTLR

ALENA KHINEIKA

FULL-STACK ENGINEER AT MONGODB

YES IT IS POSSIBLE!


MongoDB Compass Dev - localhost:27017/crunchbase.companies
My Cluster
11 DBS
22 COLLECTIONS
filter
admin
config
crunchbase
companies
db_with_collation
local
m101
mongomart
schema
test
test_read_only_user
video
localhost:27017
STANDALONE
MongoDB 3.6.2 Community
crunchbase.companies
DOCUMENTS 18.8k
TOTAL SIZE 68.9MB
AVG. SIZE 3.8KB
INDEXES 2
TOTAL SIZE 536.0KB
AVG. SIZE 268.0KB
Documents
Aggregations
Schema
ExplainPlan
Indexes
Validation
FILTER {name: 'AdventNet'}
OPTIONS
FIND
RESET
INSERT DOCUMENT
VIEW
LIST
TABLE
Displaying documents 1 - 1 of 1
_id: ObjectId("52cdef7c4bab8bd675297d8b") ObjectId
name : "AdventNet " String
permalink : "abc3 " String
crunchbase_url : "http://www.crunchbase.com/company/adventnet " String
homepage_url : "http://adventnet.com " String
blog_url : " " String
blog_feed_url : " " String
twitter_username : "manageengine " String
category_code : "enterprise " String
number_of_employees : 600 Int32
founded_year : 1996 Int32
deadpooled_year : 2 Int32
tag_list : " " String
alias_list : "Zoho ManageEngine " String
email_address : "pr@adventnet.com " String
phone_number : "925-924-9500 " String
description : "Server Management Software " String
created_at : 2007-05-25T19:24:22.000+00:00 Date
updated_at : "Wed Oct 31 18:26:09 UTC 2012 " String
: "<p>AdventNet is now Zoho ManageEngine.</p… " String
image : Object Object
products : Array Array
relationships : Array Array
competitions : Array Array
providerships : Array Array
SHOW 11 MORE FIELDS
CANCEL

Working with the mongo Shell

To display the database you are using, type **db**:

```
db
```

copy

The operation should return **test**, which is the default database.

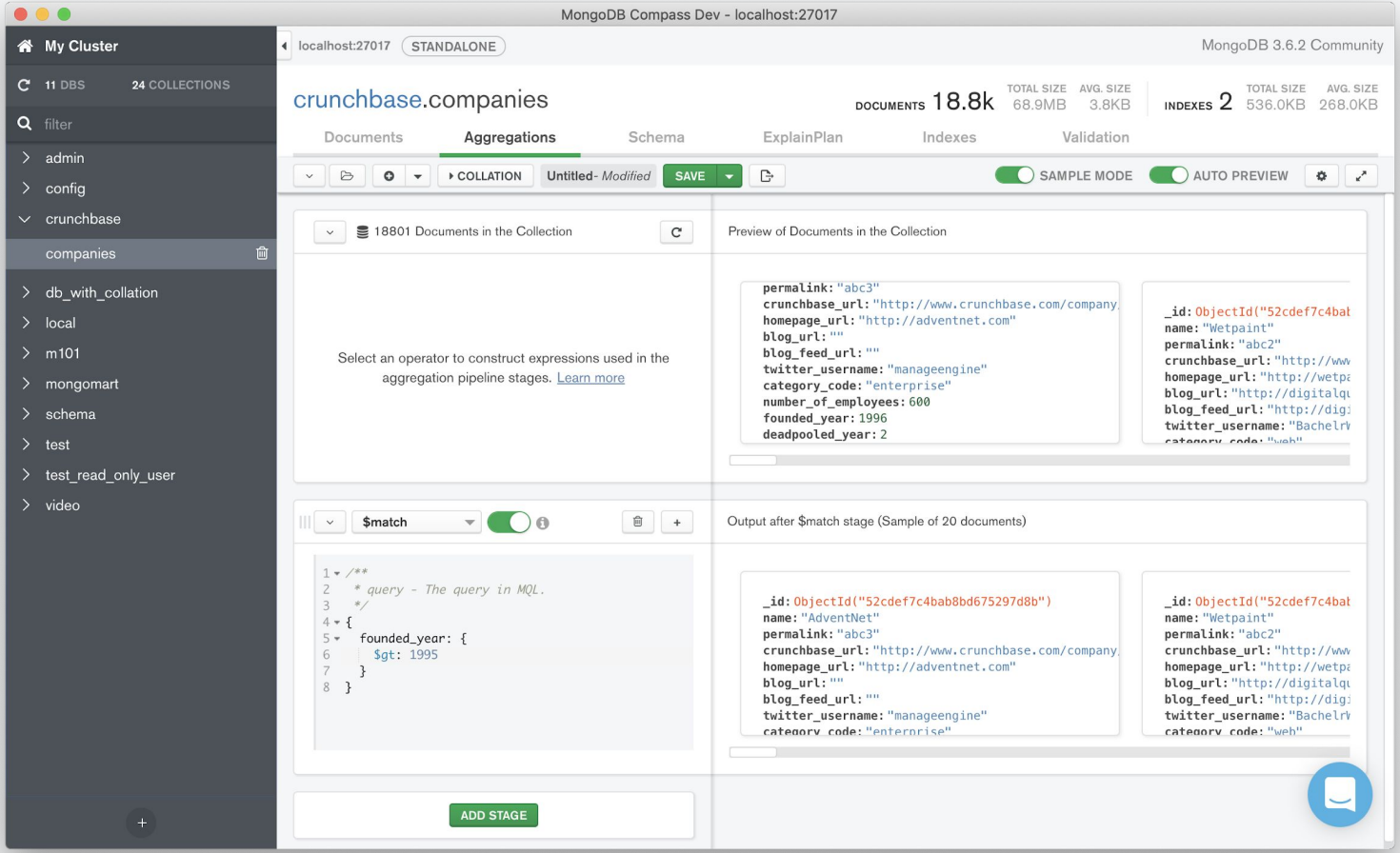
To switch databases, issue the **use <db>** helper, as in the following example:

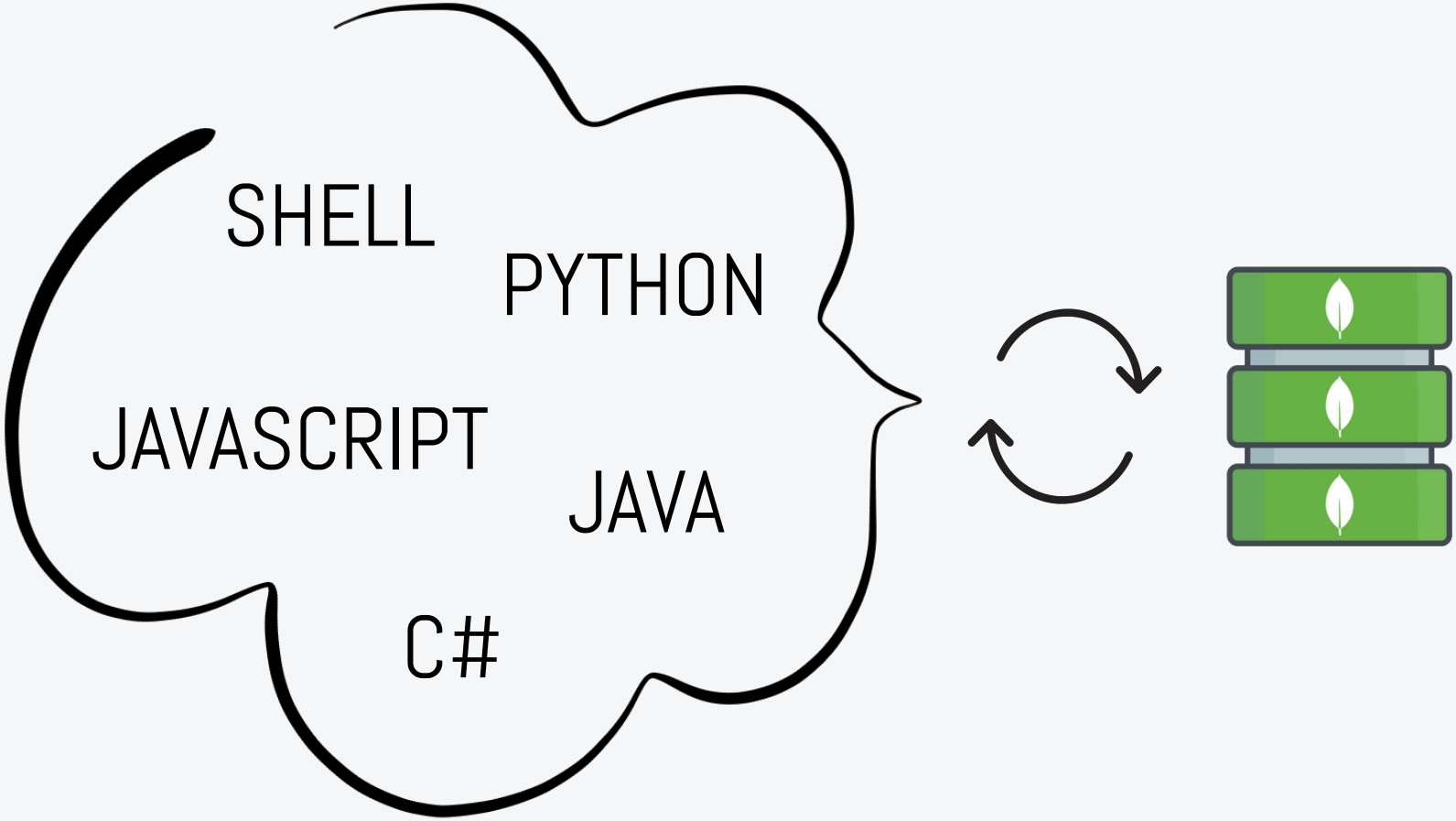
```
use <database>
```

copy

See also **db.getSiblingDB()** method to access a different database from the current database without switching your current database context (i.e. **db**).

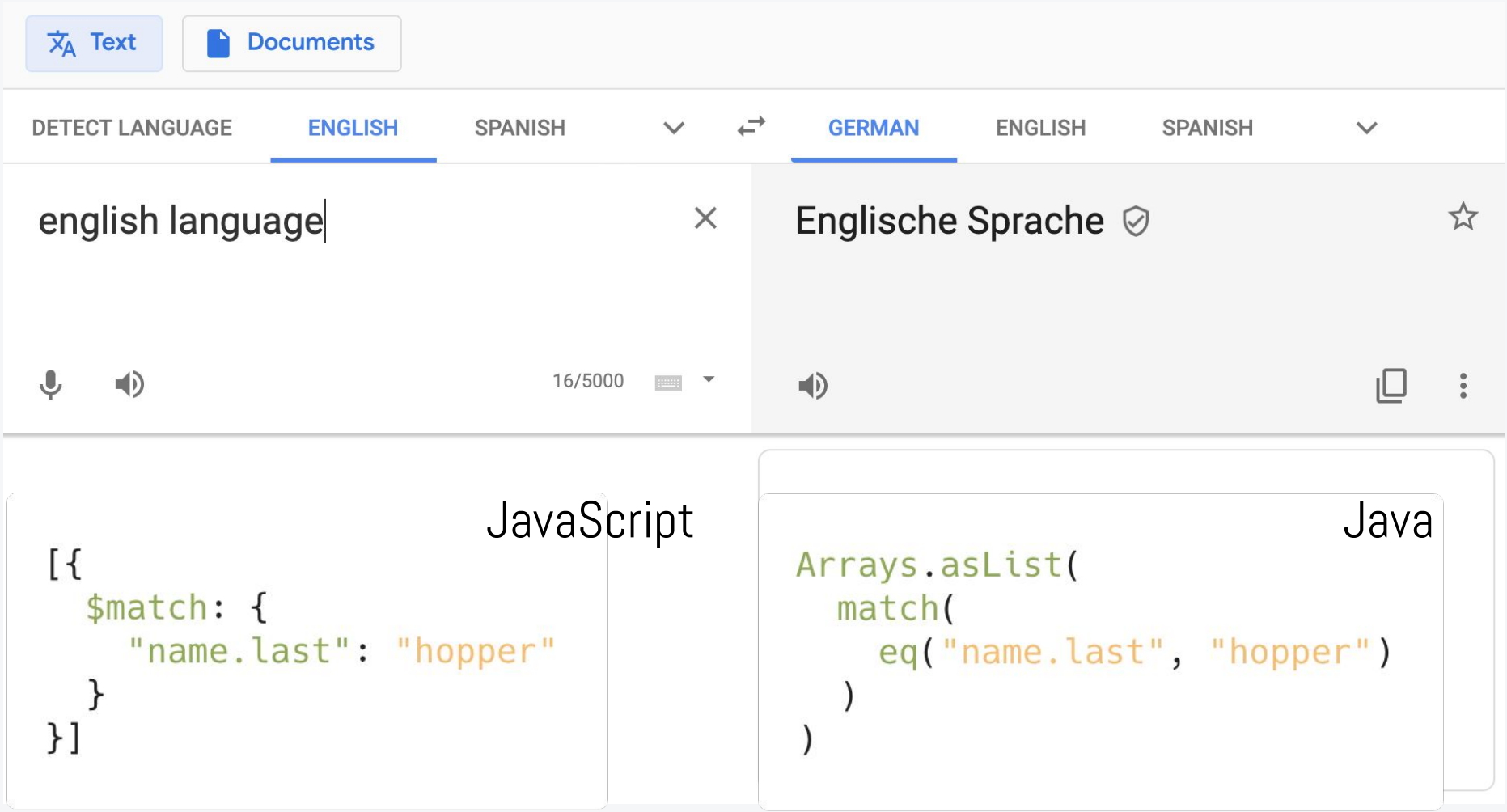
To list the databases available to the user, use the helper **show dbs**. [1]


MongoDB Compass Dev - localhost:27017
My Cluster
11 DBS
24 COLLECTIONS
filter
admin
config
crunchbase
companies
db_with_collation
local
m101
mongomart
schema
test
test_read_only_user
video
localhost:27017
STANDALONE
MongoDB 3.6.2 Community
crunchbase.companies
DOCUMENTS 18.8k
TOTAL SIZE 68.9MB
AVG. SIZE 3.8KB
INDEXES 2
TOTAL SIZE 536.0KB
AVG. SIZE 268.0KB
Documents
Aggregations
Schema
ExplainPlan
Indexes
Validation
COLLATION
Untitled- Modified
SAVE
SAMPLE MODE
AUTO PREVIEW
18801 Documents in the Collection
Preview of Documents in the Collection
Select an operator to construct expressions used in the aggregation pipeline stages. Learn more
permalink: "abc3"
crunchbase_url: "http://www.crunchbase.com/company
homepage_url: "http://adventnet.com"
blog_url: ""
blog_feed_url: ""
twitter_username: "manageengine"
category_code: "enterprise"
number_of_employees: 600
founded_year: 1996
deadpooled_year: 2
$match
Output after $match stage (Sample of 20 documents)
/**
* query - The query in MQL.
*/
{
founded_year: {
$gt: 1995
}
}
_id: ObjectId("52cdef7c4bab8bd675297d8b")
name: "AdventNet"
permalink: "abc3"
crunchbase_url: "http://www.crunchbase.com/company
homepage_url: "http://adventnet.com"
blog_url: ""
blog_feed_url: ""
twitter_username: "manageengine"
category_code: "enterprise"
_id: ObjectId("52cdef7c4bab
name: "Wetpaint"
permalink: "abc2"
crunchbase_url: "http://ww
homepage_url: "http://wetp
blog_url: "http://digitalqu
blog_feed_url: "http://digi
twitter_username: "BachelrW
category_code: "web"
ADD STAGE


SHELL
PYTHON
JAVASCRIPT
JAVA
C#

SOURCE-TO-SOURCE COMPILER

is a type of compiler that takes the source code of a program written in a programming language as its input and produces the equivalent source code in the same or a different programming language. This type of compiler is also known as transpiler.


Text
Documents
DETECT LANGUAGE
ENGLISH
SPANISH
GERMAN
ENGLISH
SPANISH
english language
16/5000
Englische Sprache
JavaScript
[{
$match: {
"name.last": "hopper"
}
}]
Java
Arrays.asList(
match(
eq("name.last", "hopper")
)
)

Query:

```
db.bios.find( { birth: { $gt: new Date('1950-01-01') } } )
```

Aggregation:

```
db.flightStats.aggregate([
  {
    $match: {
      carrierFsCode: "1I"
    }
  },
  {
    $group: {
      _id: "$departureAirportFsCode",
      flights: { $push: "$flightId" }
    }
  }
])
```

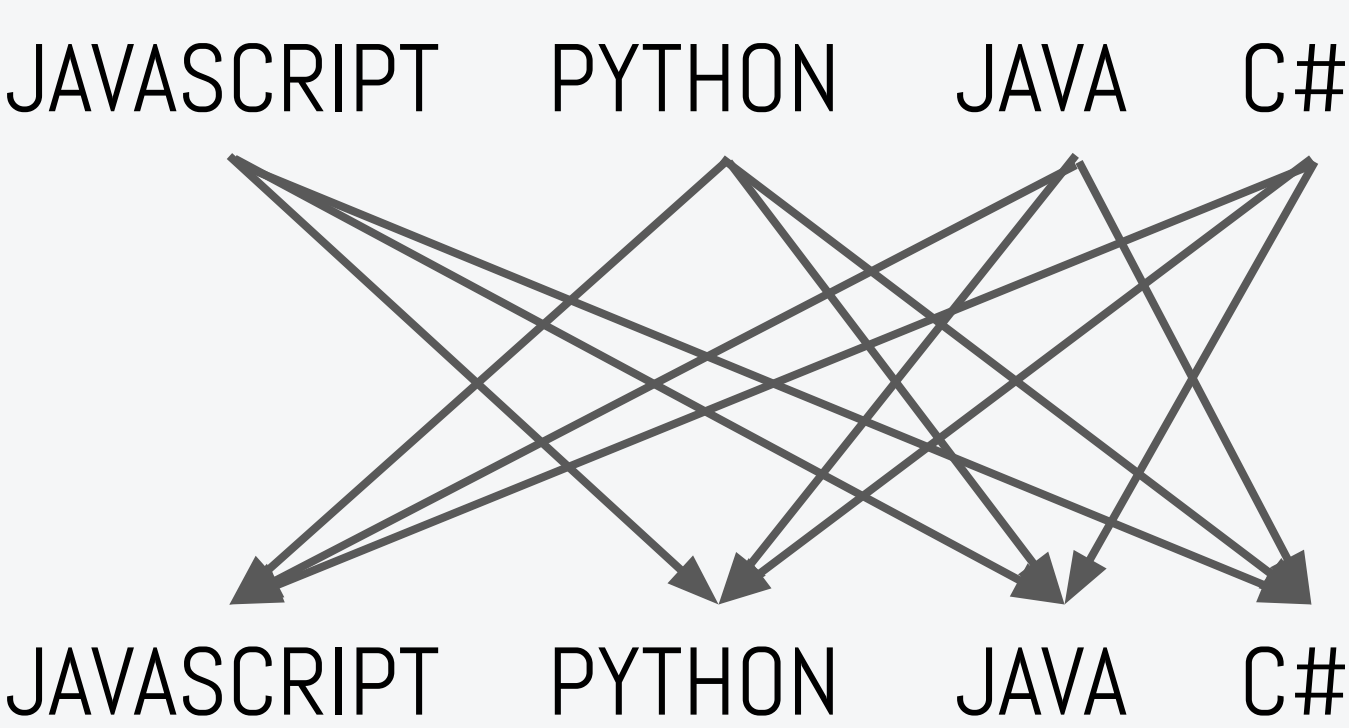
```
db.examples.aggregate([
  {
    $match: {
      start_date: new Date(date1.toISOString())
    }
  },
  {
    $group: {
      _id: { day: { $dayOfWeek: '$date' } },
      count: { $sum: '$items.quantity' }
    }
  },
  {
    $project: {
      dayOfWeek: '$_id.day',
      numberSold: '$count',
      _id: 0
    }
  }
])
```

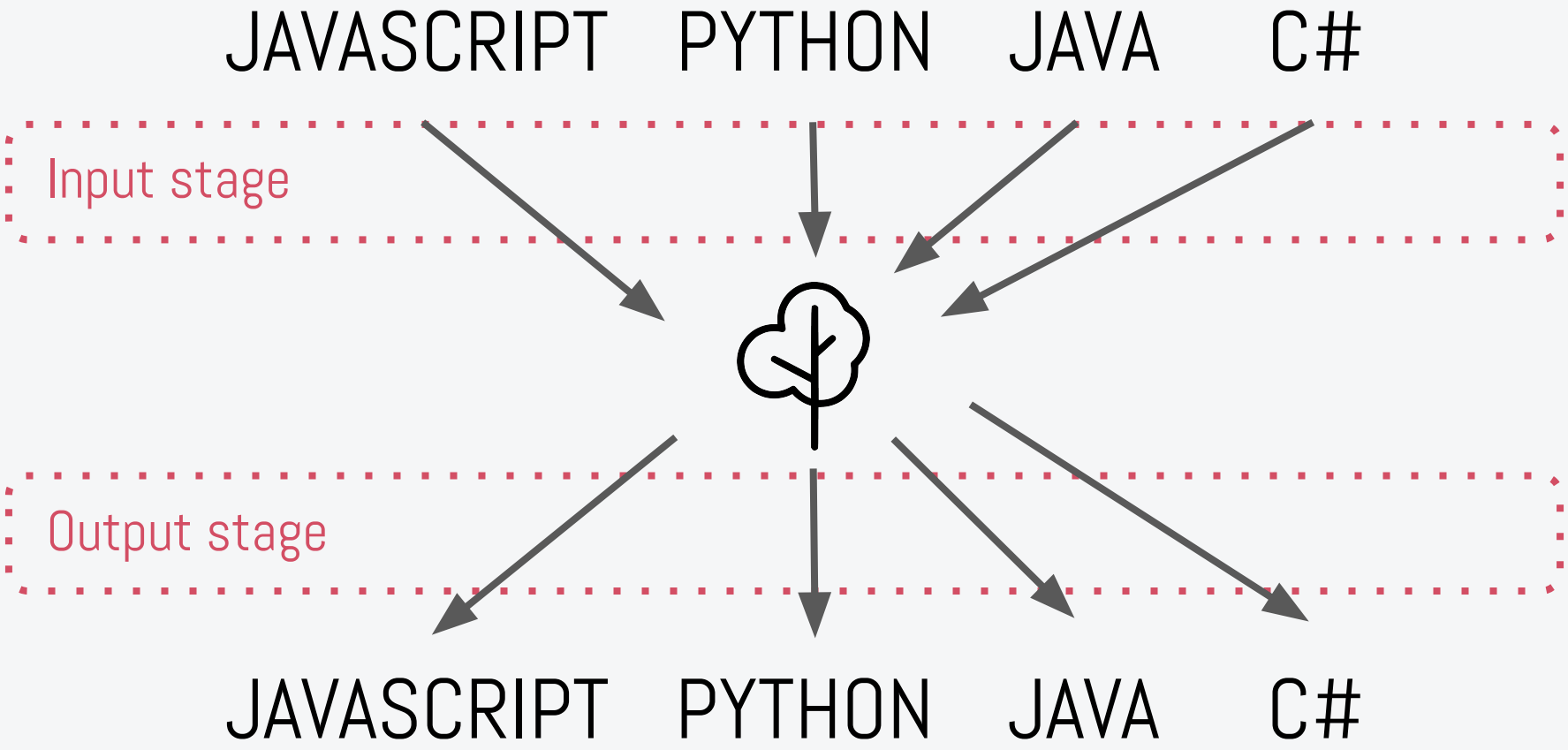
- BSON library
- Function calls
- Attributes access
- Regular expressions
- ...


node
JS

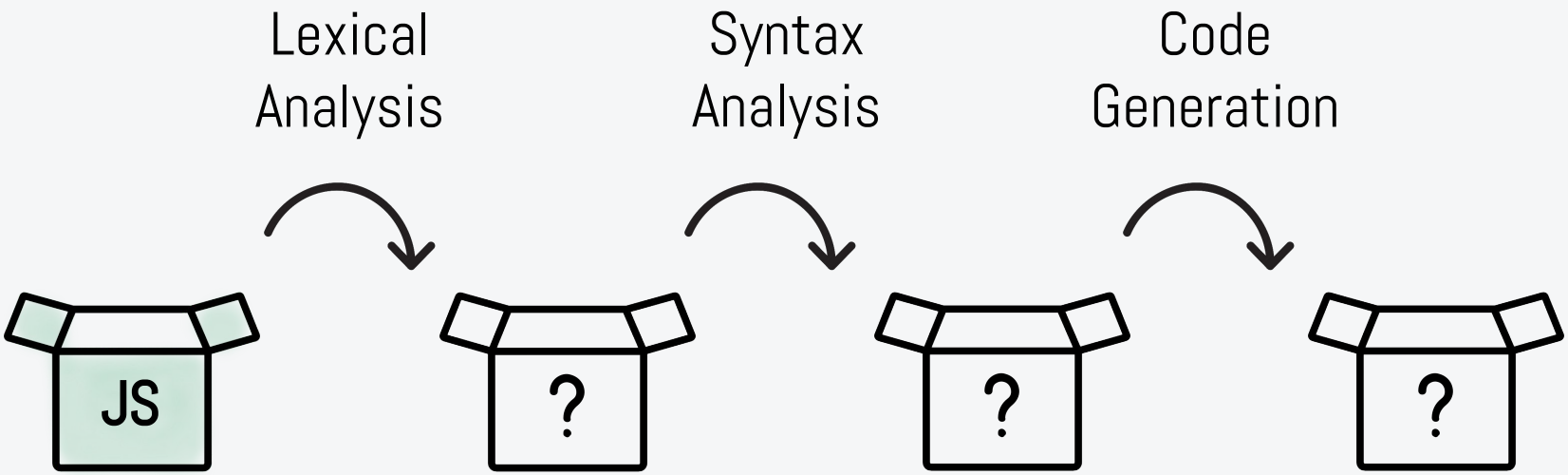
ANY LANGUAGE TO ANY LANGUAGE

\+ BSON DOCUMENTS

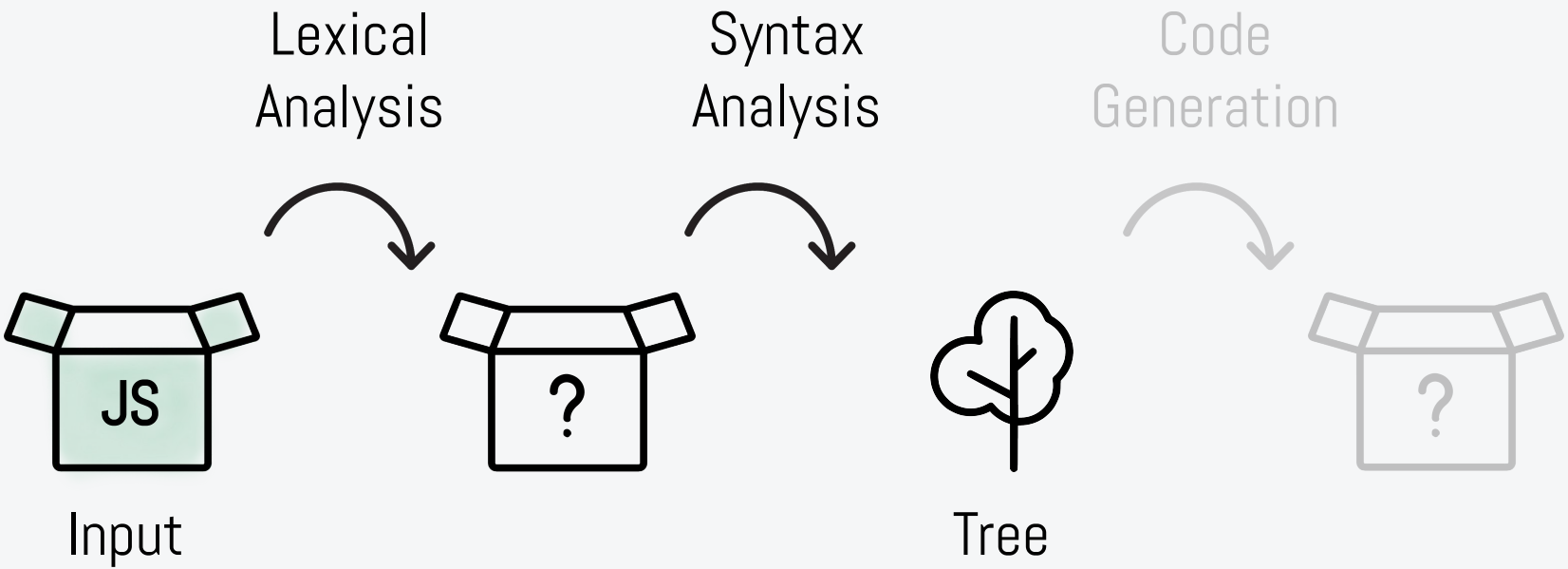

JAVASCRIPT
PYTHON
JAVA
C#
JAVASCRIPT
PYTHON
JAVA
C#


JAVASCRIPT PYTHON JAVA C#
Input stage
Output stage
JAVASCRIPT PYTHON JAVA C#

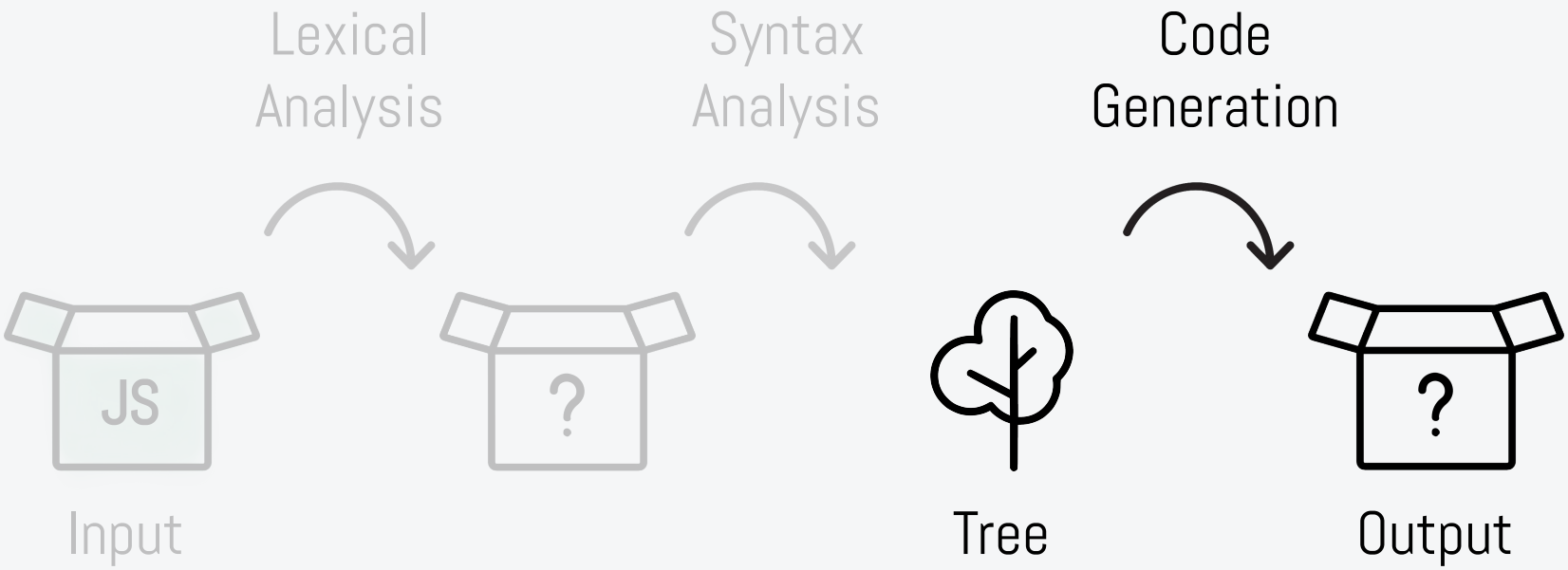
3 STAGES OF COMPILER DESIGN



INPUT STAGE


Lexical Analysis
Syntax Analysis
Code Generation
JS
?
?
Input
Tree

OUTPUT STAGE



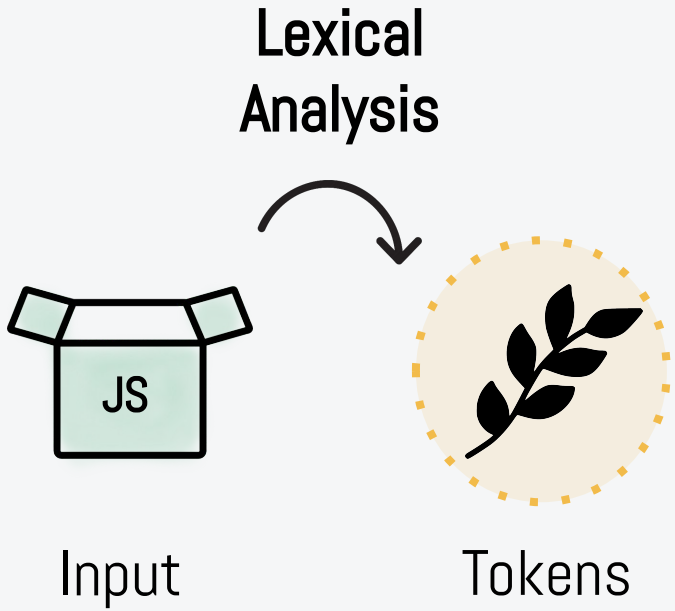
1. LEXICAL ANALYSIS

Raw Code

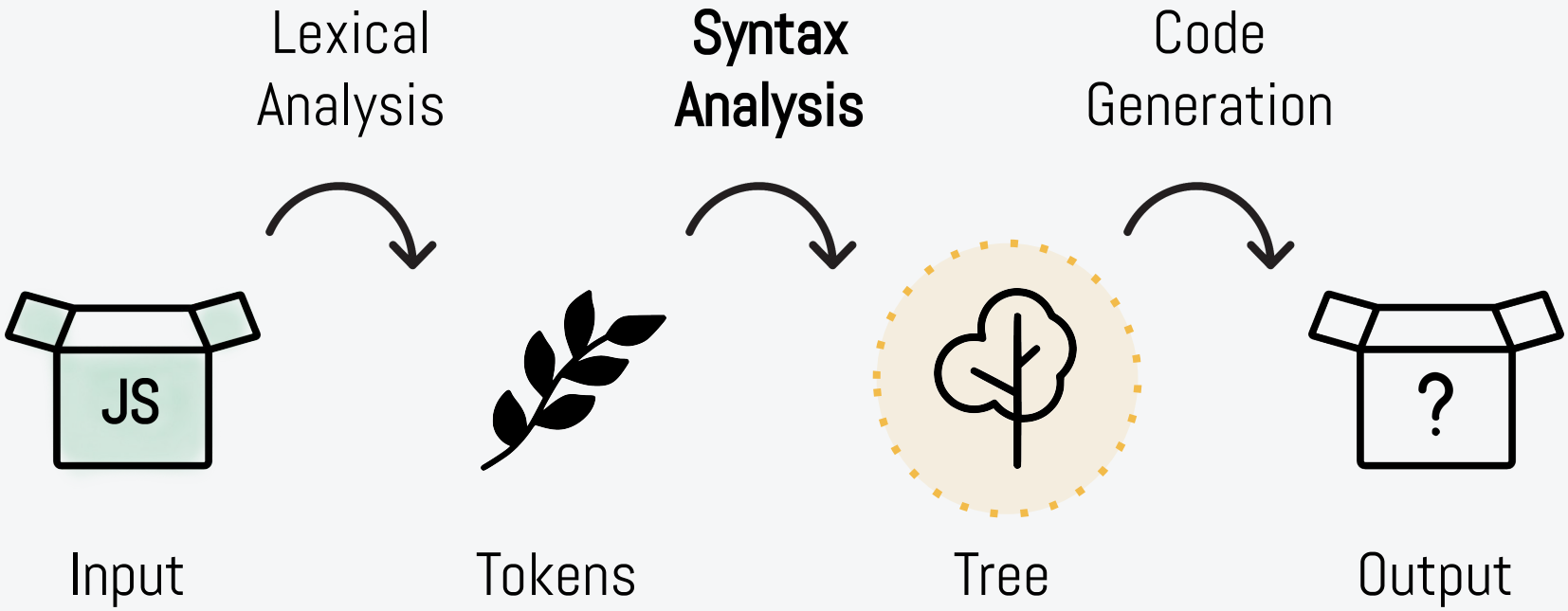
{ x : 1 }

Tokens

OpenBrace	{
Identifier	x
Colon	:
Identifier	1
CloseBrace	}



2. SYNTAX ANALYSIS



FROM STRING TO TREE

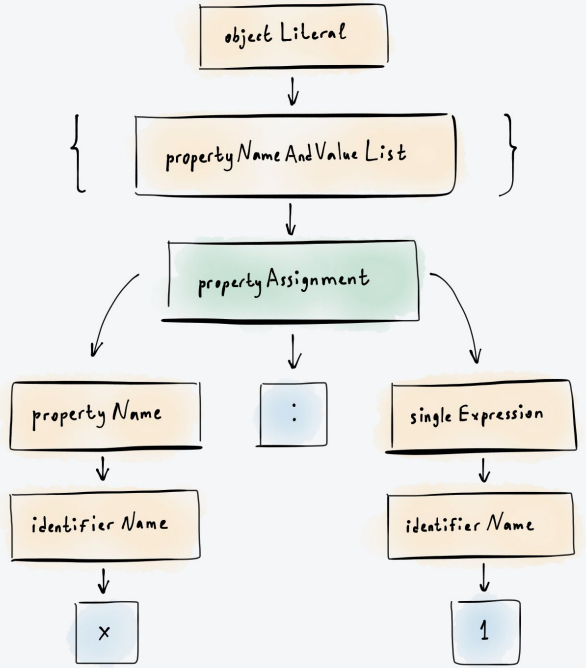
Raw Code

{ X : 1 }

Tokens

OpenBrace	{
Identifier	x
Colon	:
Identifier	1
CloseBrace	}

Tree Structure

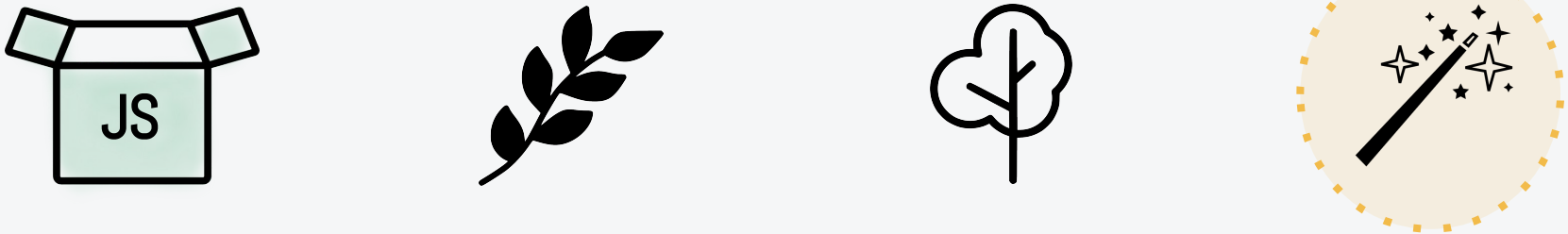


3. CODE GENERATION

Lexical Analysis

Syntax Analysis

Code Generation



Input

Tokens

Tree

Output


Don't write
from scratch!

```
/// PropertyNameAndValueList :
///     PropertyAssignment
///     PropertyNameAndValueList , PropertyAssignment
propertyNameAndValueList
 : propertyAssignment ( ',' propertyAssignment )*
 ;

/// PropertyAssignment :
///     PropertyName : AssignmentExpression
///     get PropertyName ( ) { FunctionBody }
///     set PropertyName ( PropertySetParameterList ) { FunctionBody }
propertyAssignment
 : propertyName ':' singleExpression                          # PropertyExpressionAssignment
 | getter '(' ')' '{' functionBody '}'                        # PropertyGetter
 | setter '(' propertySetParameterList ')' '{' functionBody '}' # PropertySetter
 ;

/// PropertyName :
///     IdentifierName
///     StringLiteral
///     NumericLiteral
propertyName
 : identifierName
 | StringLiteral
 | numericLiteral
 ;
```

```
412   /// PropertyNameAndValueList :
413   ///     PropertyAssignment
414   ///     PropertyNameAndValueList , PropertyAssignment
415   propertyNameAndValueList
416    : propertyAssignment ( ',' propertyAssignment )*
417    ;
418
419   /// PropertyAssignment :
420   ///     PropertyName : AssignmentExpression
421   ///     get PropertyName ( ) { FunctionBody }
422   ///     set PropertyName ( PropertySetParameterList ) { FunctionBody }
423   propertyAssignment
424    : propertyName ':' singleExpression                          # PropertyExpressionAssignment
425    | getter '(' ')' '{' functionBody '}'                        # PropertyGetter
426    | setter '(' propertySetParameterList ')' '{' functionBody '}' # PropertySetter
427    ;
428
429   /// PropertyName :
430   ///     IdentifierName
431   ///     StringLiteral
432   ///     NumericLiteral
433   propertyName
434    : identifierName
435    | StringLiteral
436    | numericLiteral
437    ;
```

```
412  /// PropertyNameAndValueList :
413  ///     PropertyAssignment
414  ///     PropertyNameAndValueList , PropertyAssignment
415  propertyNameAndValueList
416   : propertyAssignment ( ',' propertyAssignment )*
417   ;
418
419  /// PropertyAssignment :
420  ///     PropertyName : AssignmentExpression
421  ///     get PropertyName ( ) { FunctionBody }
422  ///     set PropertyName ( PropertySetParameterList ) { FunctionBody }
423  propertyAssignment
424   : propertyName ':' singleExpression                # PropertyExpressionAssignment
425   | getter '(' ')' '{' functionBody '}'              # PropertyGetter
426   | setter '(' propertySetParameterList ')' '{' functionBody '}' # PropertySetter
427   ;
428
429  /// PropertyName :
430  ///     IdentifierName
431  ///     StringLiteral
432  ///     NumericLiteral
433  propertyName
434   : identifierName
435   | StringLiteral
436   | numericLiteral
437   ;
```

* ANTLR - Another Tool for Language Recognition

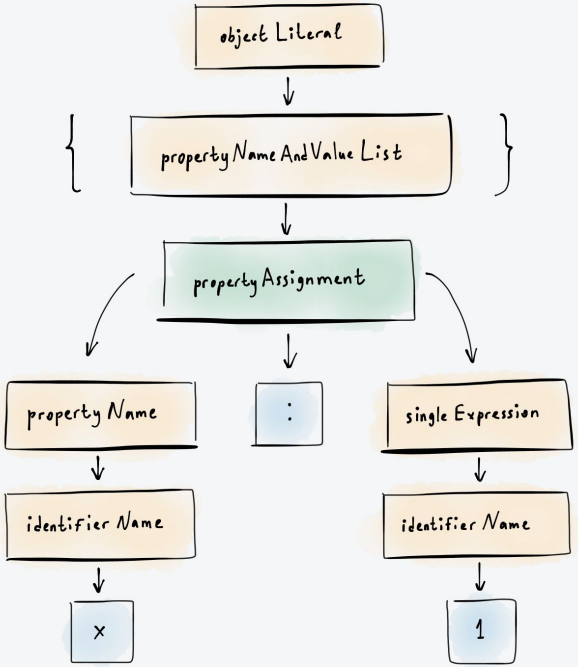
- Provides grammars
- Has a JavaScript runtime
- Generates a parser
- Builds a parse tree

PARSE TREE VS. AST

* AST - Abstract Syntax Tree

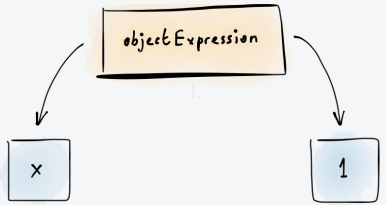
Parse Tree

- Represents concrete syntax
- Contain unusable information



Abstract Syntax Tree

- Represents abstract syntax
- Contain only meaningful information



VISITOR VS. LISTENER

Visitor

- You should explicitly call child methods
- Methods can return any custom type
- You can modify existing nodes

```
↓ visitPropertyNameAndValue(ctx)
↓ visitPropertyAssignment(ctx)
↓ visitPropertyName(ctx)
↓ visitIdentifierName(ctx)
↓ visitTerminal(ctx)
```

Listener

- Methods are called automatically
- Methods can't return a value
- You should store values outside the tree

```
→ enterPropertyAssignment(ctx)
  → enterPropertyName(ctx)
    → enterIdentifierName(ctx)
      → enterTerminal(ctx)
      ← exitTerminal(ctx)
    ← exitIdentifierName(ctx)
  ← exitPropertyName(ctx)
← exitPropertyAssignment(ctx)
```

VISITOR

PARSE TREE

ANTLR AUXILIARY FILES

*Lexer.js
Lexical Analysis

*Parser.js
Syntax Analysis

*Visitor.js
Tree Traversal

Input

Characters

Tokens

Tree

Output

JS ECMAScriptVisitor.js

```
var antlr4 = require('antlr4/index');

function ECMAScriptVisitor() {
  antlr4.tree.ParseTreeVisitor.call(this);
  return this;
}

ECMAScriptVisitor.prototype = Object.create(antlr4.tree.ParseTreeVisitor.prototype);
ECMAScriptVisitor.prototype.constructor = ECMAScriptVisitor;

ECMAScriptVisitor.prototype.visitProgram = function(ctx) {
  return this.visitChildren(ctx);
};

ECMAScriptVisitor.prototype.visitPropertyExpressionAssignment = function(ctx) {
  return this.visitChildren(ctx);
};
```

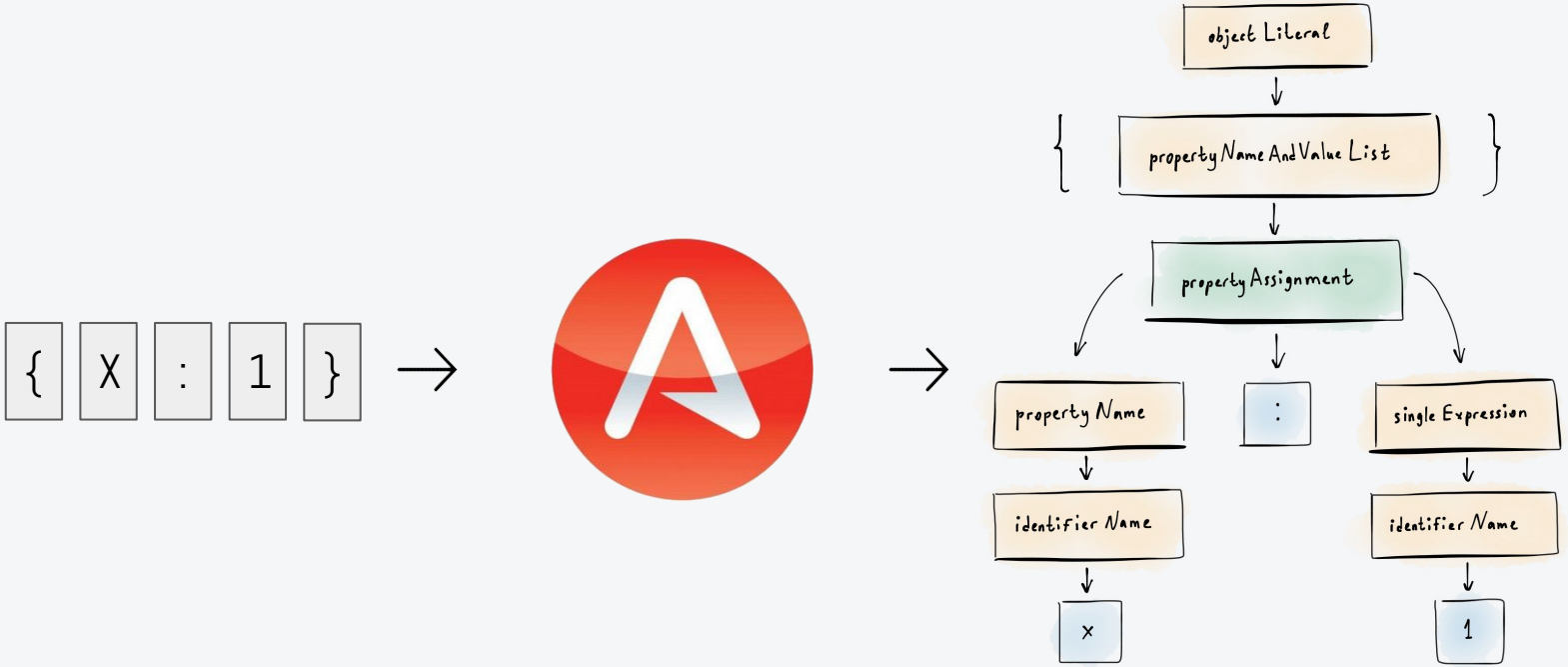
Not here!

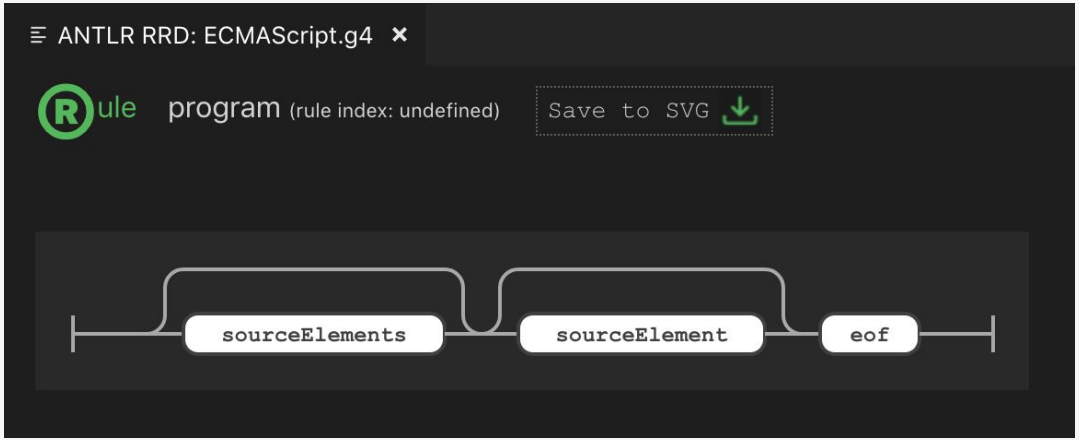
JS PythonGenerator.js

```
const ECMAScriptVisitor = require('../lib/ECMAScriptVisitor').ECMAScriptVisitor;

class Visitor extends ECMAScriptVisitor {}

module.exports = Visitor;
```

Here!

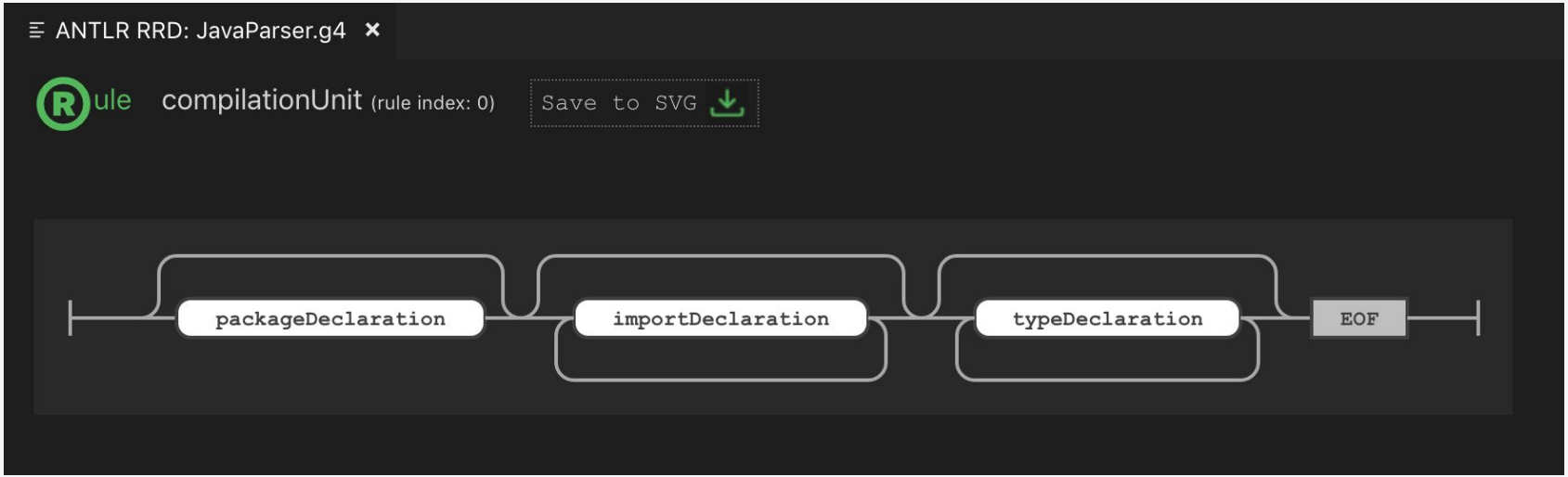
PARSE TREE


{
X
:
1
}
objectLiteral
{
propertyNameAndValueList
}
propertyAssignment
propertyName
:
singleExpression
identifierName
identifierName
x
1



JavaScript

Java



JS index.js ×

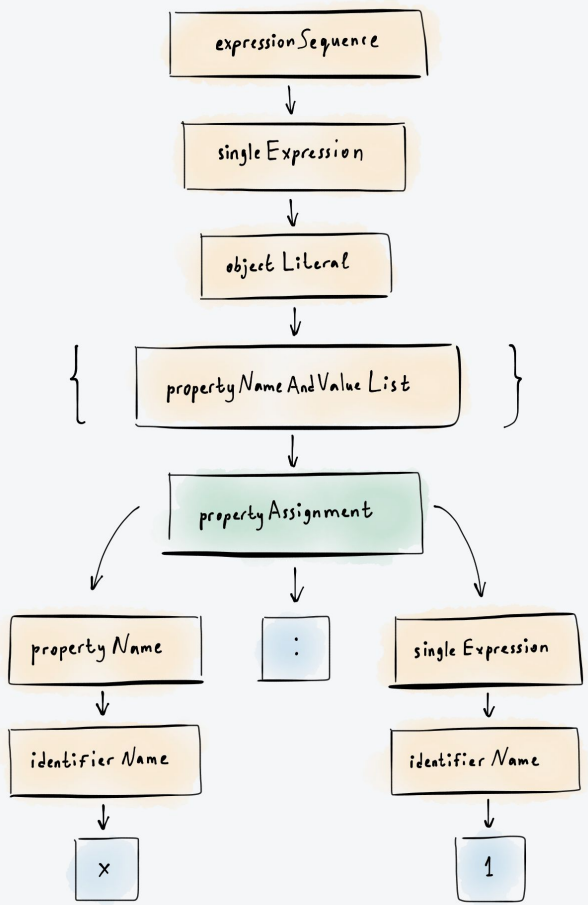
```
1   const antlr4 = require('antlr4');
2   const ECMAScriptLexer = require('./lib/ECMAScriptLexer.js');
3   const ECMAScriptParser = require('./lib/ECMAScriptParser.js');
4
5   const input = '{x: 1}';
6
7   const chars = new antlr4.InputStream(input);
8   const lexer = new ECMAScriptLexer.ECMAScriptLexer(chars);
9   const tokens = new antlr4.CommonTokenStream(lexer);
10  const parser = new ECMAScriptParser.ECMAScriptParser(tokens);
11
12  const tree = parser.expressionSequence();
```

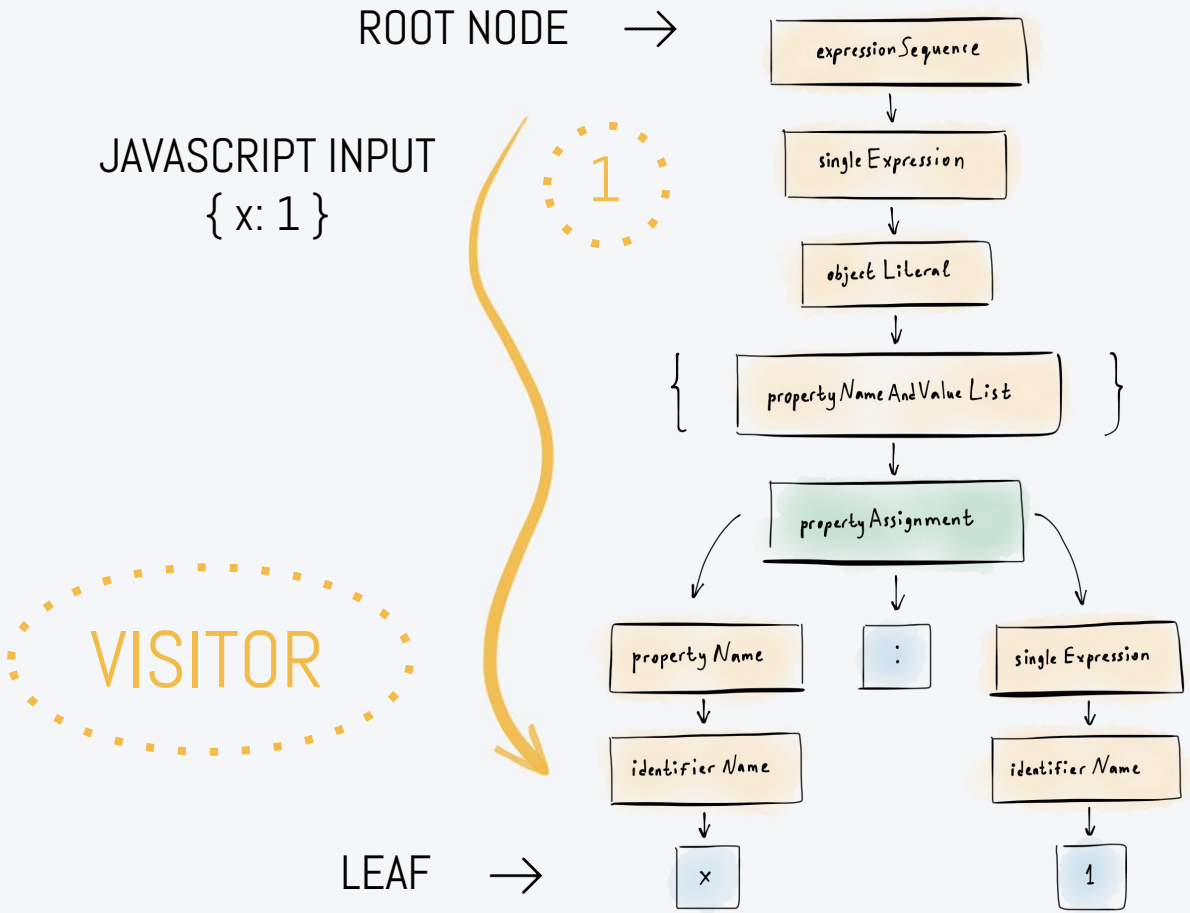
ECMAScript.interp

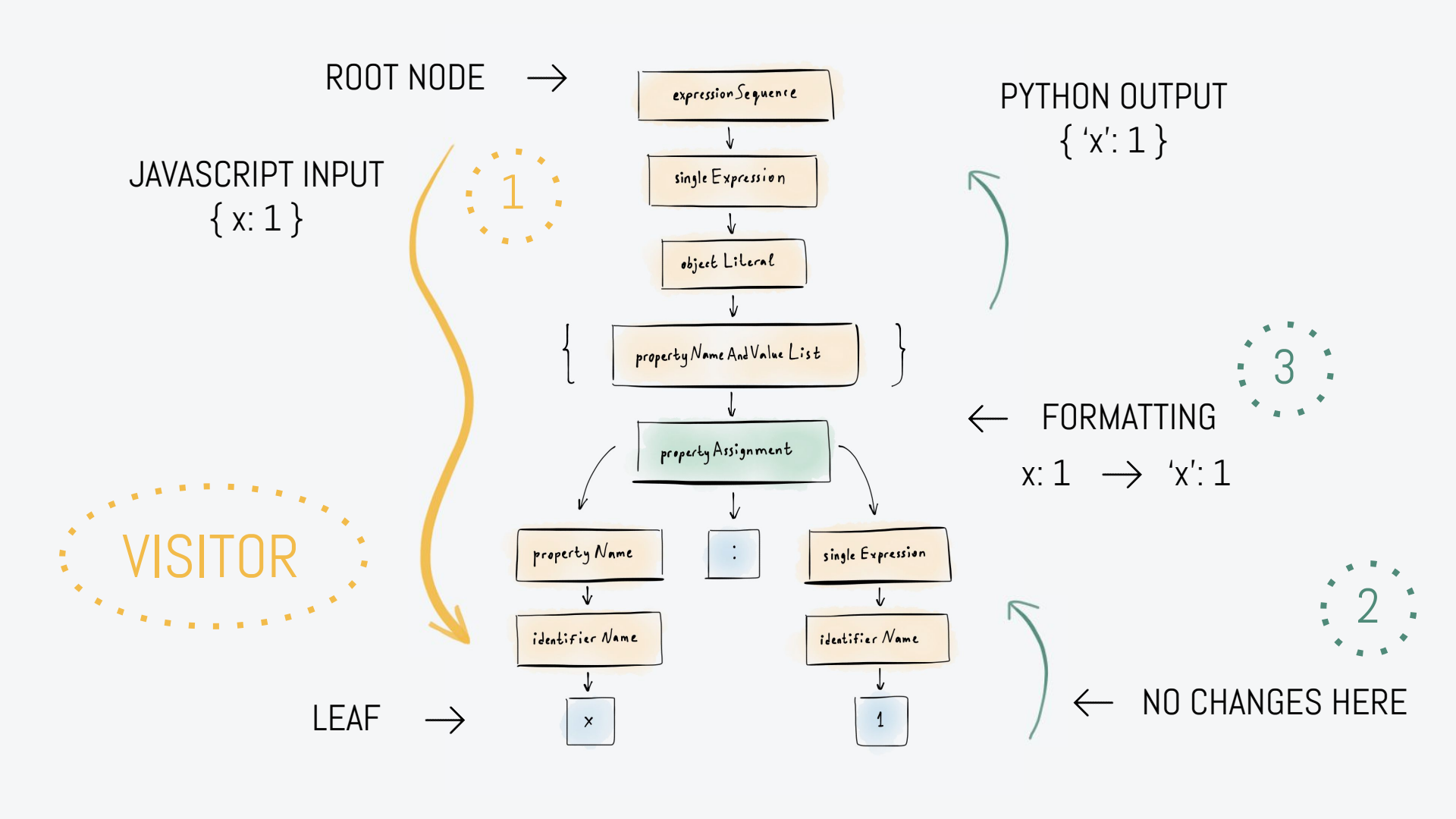
```
rule names:
program
sourceElements
sourceElement
statement
block
statementList
variableStatement
variableDeclarationList
variableDeclaration
initialiser
emptyStatement
expressionStatement
ifStatement
```

index.js

```
const chars = new antlr4.InputStream(input);
const lexer = new ECMAScriptLexer.ECMAScriptLexer(chars);
const tokens = new antlr4.CommonTokenStream(lexer);
const parser = new ECMAScriptParser.ECMAScriptParser(tokens);

const tree = parser.expressionSequence();
```




ROOT NODE
JAVASCRIPT INPUT
{ x: 1 }
1
VISITOR
LEAF
expressionSequence
singleExpression
objectLiteral
{
propertyNameAndValueList
}
propertyAssignment
propertyName
:
singleExpression
identifierName
identifierName
x
1


ROOT NODE
expressionSequence
singleExpression
objectLiteral
{
propertyNameAndValueList
}
propertyAssignment
propertyName
:
singleExpression
identifierName
identifierName
x
1
LEAF
JAVASCRIPT INPUT
{ x: 1 }
1
PYTHON OUTPUT
{ 'x': 1 }
3
FORMATTING
x: 1 → 'x': 1
2
NO CHANGES HERE
VISITOR

JS PythonGenerator.js ✕

```
1  const ECMAScriptVisitor = require('../lib/ECMAScriptVisitor').ECMAScriptVisitor;
2
3  class Visitor extends ECMAScriptVisitor {
4    start(ctx) {
5      return this.visitExpressionSequence(ctx);
6    }
7
8
9
10
11 }
12
13 module.exports = Visitor;
14
```

JS PythonGenerator.js ✕

```
1  const ECMAScriptVisitor = require('../lib/ECMAScriptVisitor').ECMAScriptVisitor;
2
3  class Visitor extends ECMAScriptVisitor {
4    start(ctx) {
5      return this.visitExpressionSequence(ctx);
6    }
7
8    visitTerminal(ctx) {
9      return ctx.getText();
10   }
11 }
12
13 module.exports = Visitor;
14
```

SINGLE AND DOUBLE QUOTES

What do we support?

DIFFERENT NUMERAL SYSTEMS

Answer should be accurate

USER INPUT FORMAT

Spaces, empty lines etc.

ESCAPE SEQUENCES

Have to be escaped

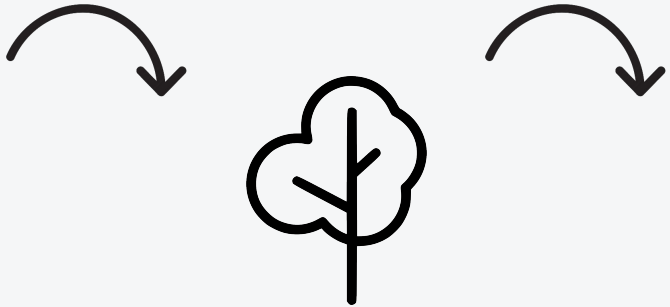
FLOATING POINT NUMBERS

Don't damage Pi

COMMENTS

Yes / No

JAVASCRIPT
{ x: 1 }



PYTHON
{ 'x': 1 }

JS PythonGenerator.js ●

```
const ECMAScriptVisitor = require('../lib/ECMAScriptVisitor')
  .ECMAScriptVisitor;

class Visitor extends ECMAScriptVisitor {
  start(ctx) {
    return this.visitExpressionSequence(ctx);
  }

  visitPropertyExpressionAssignment(ctx) {
    const key = this.visit(ctx.propertyName());
    const value = this.visit(ctx.singleExpression());

    return `'${key}': ${value}`;
  }

  visitTerminal(ctx) {
    return ctx.getText();
  }
}

module.exports = Visitor;
```

JS PythonGenerator.js

```
const ECMAScriptVisitor = require('../lib/ECMAScriptVisitor')
  .ECMAScriptVisitor;

class Visitor extends ECMAScriptVisitor {
  start(ctx) {
    return this.visitExpressionSequence(ctx);
  }

  visitPropertyExpressionAssignment(ctx) {
    const key = this.visit(ctx.propertyName());
    const value = this.visit(ctx.singleExpression());

    return `'${key}': ${value}`;
  }

  visitTerminal(ctx) {
    return ctx.getText();
  }
}

module.exports = Visitor;
```

Formatting

x: 1 → 'x': 1

```
608     | singleExpression '=' expressionSequence                 # AssignmentExpression
609     | singleExpression assignmentOperator expressionSequence  # AssignmentOperatorExpression
610     | This                                                    # ThisExpression
611     | Identifier                                              # IdentifierExpression
612     | Number arguments                                        # NumberExpression
613     | literal                                                 # LiteralExpression
614     | arrayLiteral                                            # ArrayLiteralExpression
615     | objectLiteral                                           # ObjectLiteralExpression
616     | '(' expressionSequence ')'                              # ParenthesizedExpression
617     ;
...
835 Function   : 'function';
836 This       : 'this';
837 With       : 'with';
838 Default    : 'default';
839 If         : 'if';
840 Throw      : 'throw';
841 Delete     : 'delete';
842 In         : 'in';
843 Try        : 'try';
844 Number     : 'Number';
845
846 /// 7.6.1.2 Future Reserved Words
847 Class   : 'class';
```

```
608     | singleExpression '=' expressionSequence                 # AssignmentExpression
609     | singleExpression assignmentOperator expressionSequence  # AssignmentOperatorExpression
610     | This                                                    # ThisExpression
611     | Identifier                                              # IdentifierExpression
612     | Number arguments                                        # NumberExpression
613     | literal                                                 # LiteralExpression
614     | arrayLiteral                                            # ArrayLiteralExpression
615     | objectLiteral                                           # ObjectLiteralExpression
616     | '(' expressionSequence ')'                              # ParenthesizedExpression
617     ;
...
835 Function   : 'function';
836 This       : 'this';
837 With       : 'with';
838 Default    : 'default';
839 If         : 'if';
840 Throw      : 'throw';
841 Delete     : 'delete';
842 In         : 'in';
843 Try        : 'try';
844 Number     : 'Number';
845
846 /// 7.6.1.2 Future Reserved Words
847 Class   : 'class';
```

```
608     | singleExpression '=' expressionSequence                  # AssignmentExpression
609     | singleExpression assignmentOperator expressionSequence   # AssignmentOperatorExpression
610     | This                                                     # ThisExpression
611     | Identifier                                               # IdentifierExpression
612     | Number arguments                                         # NumberExpression
613     | literal                                                  # LiteralExpression
614     | arrayLiteral                                             # ArrayLiteralExpression
615     | objectLiteral                                            # ObjectLiteralExpression
616     | '(' expressionSequence ')'                               # ParenthesizedExpression
617     ;
...
835   Function   : 'function';
836   This       : 'this';
837   With       : 'with';
838   Default    : 'default';
839   If         : 'if';
840   Throw      : 'throw';
841   Delete     : 'delete';
842   In         : 'in';
843   Try        : 'try';
844   Number     : 'Number';
845
846   /// 7.6.1.2 Future Reserved Words
847   Class    : 'class';
```

```
608     | singleExpression '=' expressionSequence                  # AssignmentExpression
609     | singleExpression assignmentOperator expressionSequence   # AssignmentOperatorExpression
610     | This                                                     # ThisExpression
611     | Identifier                                               # IdentifierExpression
612     | Number arguments                                         # NumberExpression
613     | literal                                                  # LiteralExpression
614     | arrayLiteral                                             # ArrayLiteralExpression
615     | objectLiteral                                            # ObjectLiteralExpression
616     | '(' expressionSequence ')'                               # ParenthesizedExpression
617     ;
...
835 Function   : 'function';
836 This       : 'this';
837 With       : 'with';
838 Default    : 'default';
839 If         : 'if';
840 Throw      : 'throw';
841 Delete     : 'delete';
842 In         : 'in';
843 Try        : 'try';
844 Number     : 'Number';
845
846 /// 7.6.1.2 Future Reserved Words
847 Class   : 'class';
```

JS PythonGenerator.js ●

```
8
9     visitPropertyExpressionAssignment(ctx) {
10      const key = this.visit(ctx.propertyName());
11      const value = this.visit(ctx.singleExpression());
12
13      return `'${key}': ${value}`;
14    }
15
16    visitTerminal(ctx) {
17      return ctx.getText();
18    }
19
20    visitNumberExpression(ctx) {
21      const argumentList = ctx.arguments().argumentList();
22      const arg = argumentList.singleExpression()[0];
23      const number = this.visit(arg);
24
25      return `int(${number})`;
26    }
27  }
28
29  module.exports = Visitor;
30
```

JavaScript input:

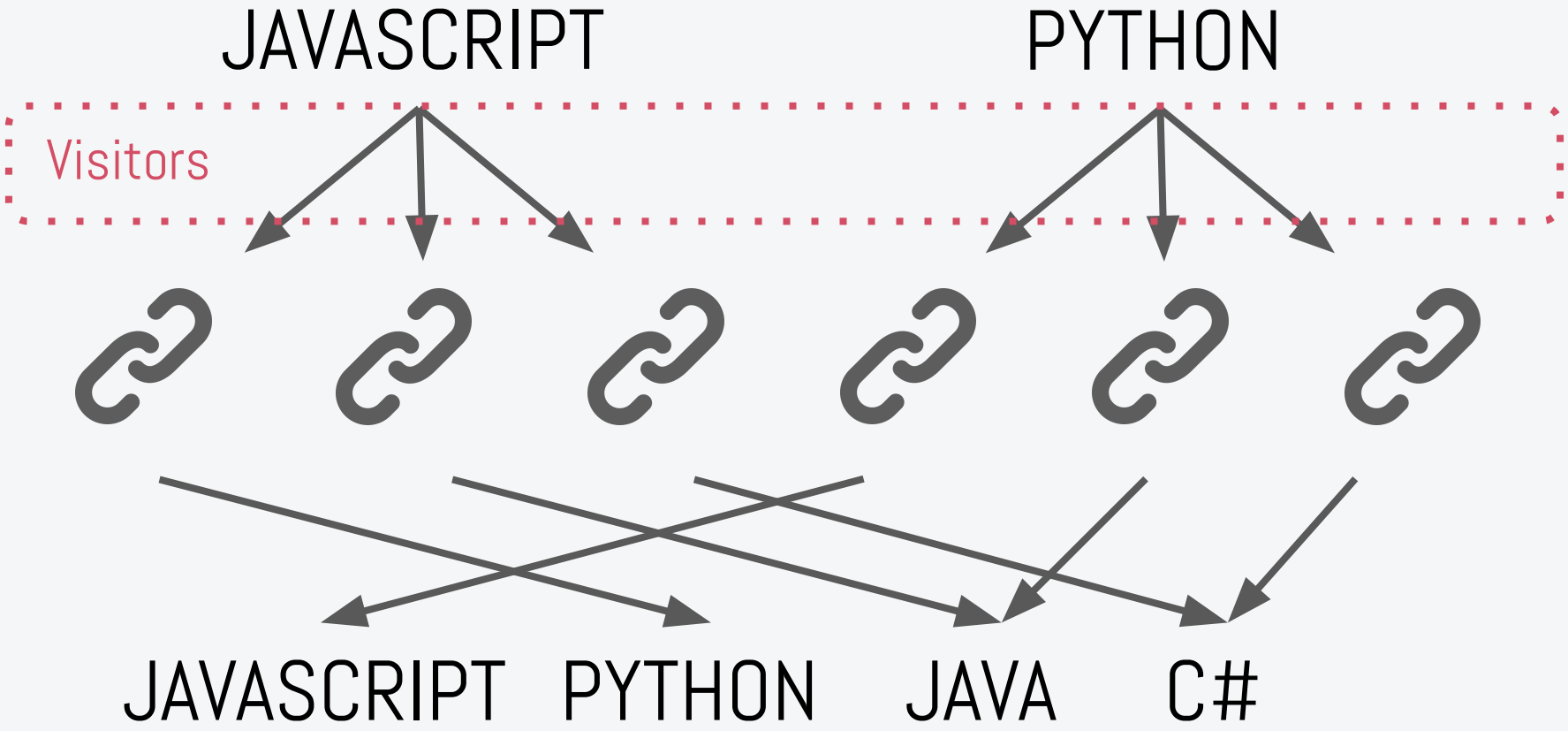
{ x: Number(1) }

Python output:

{ 'x': int(1) }

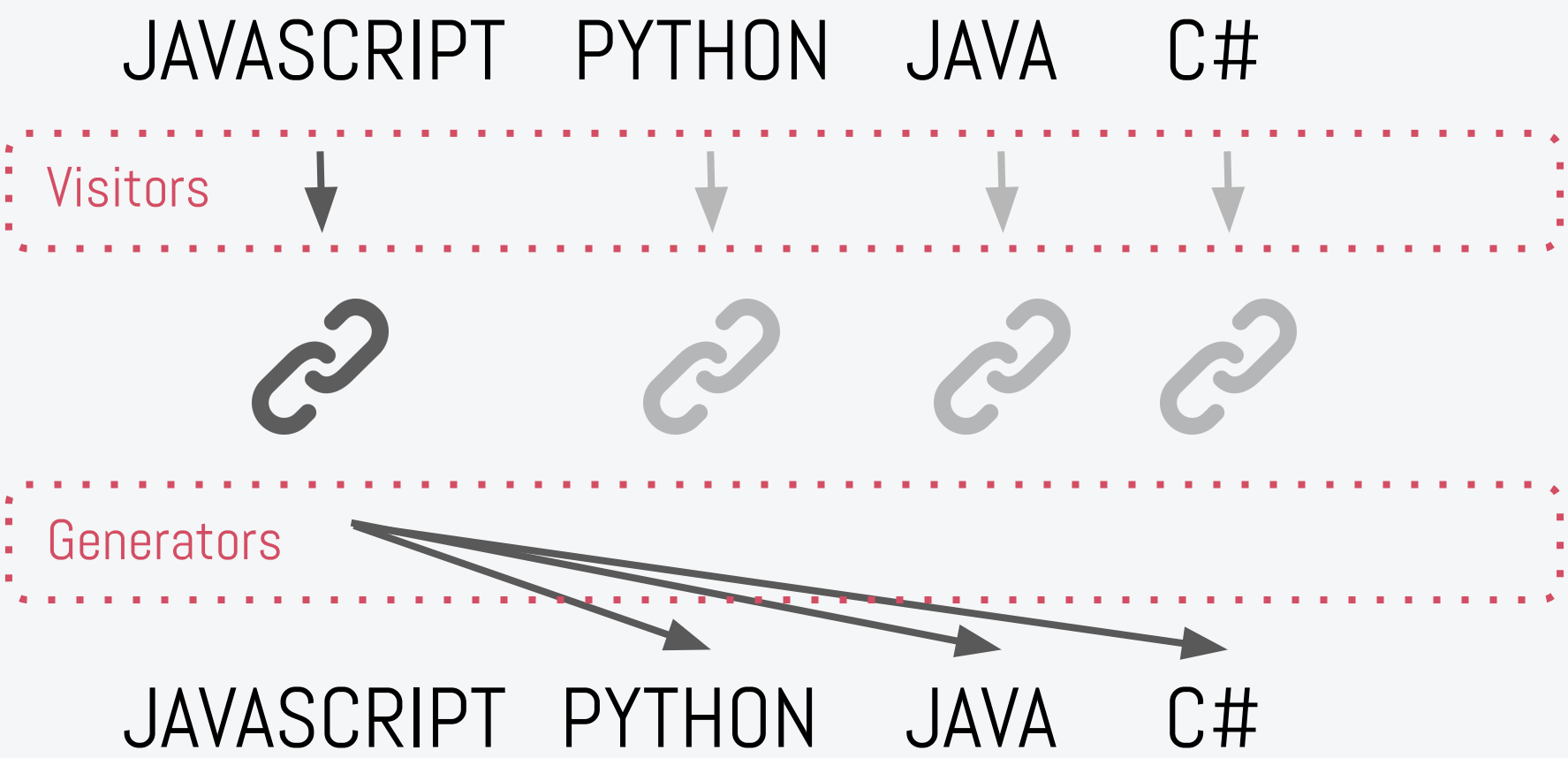
JS PythonGenerator.js ●

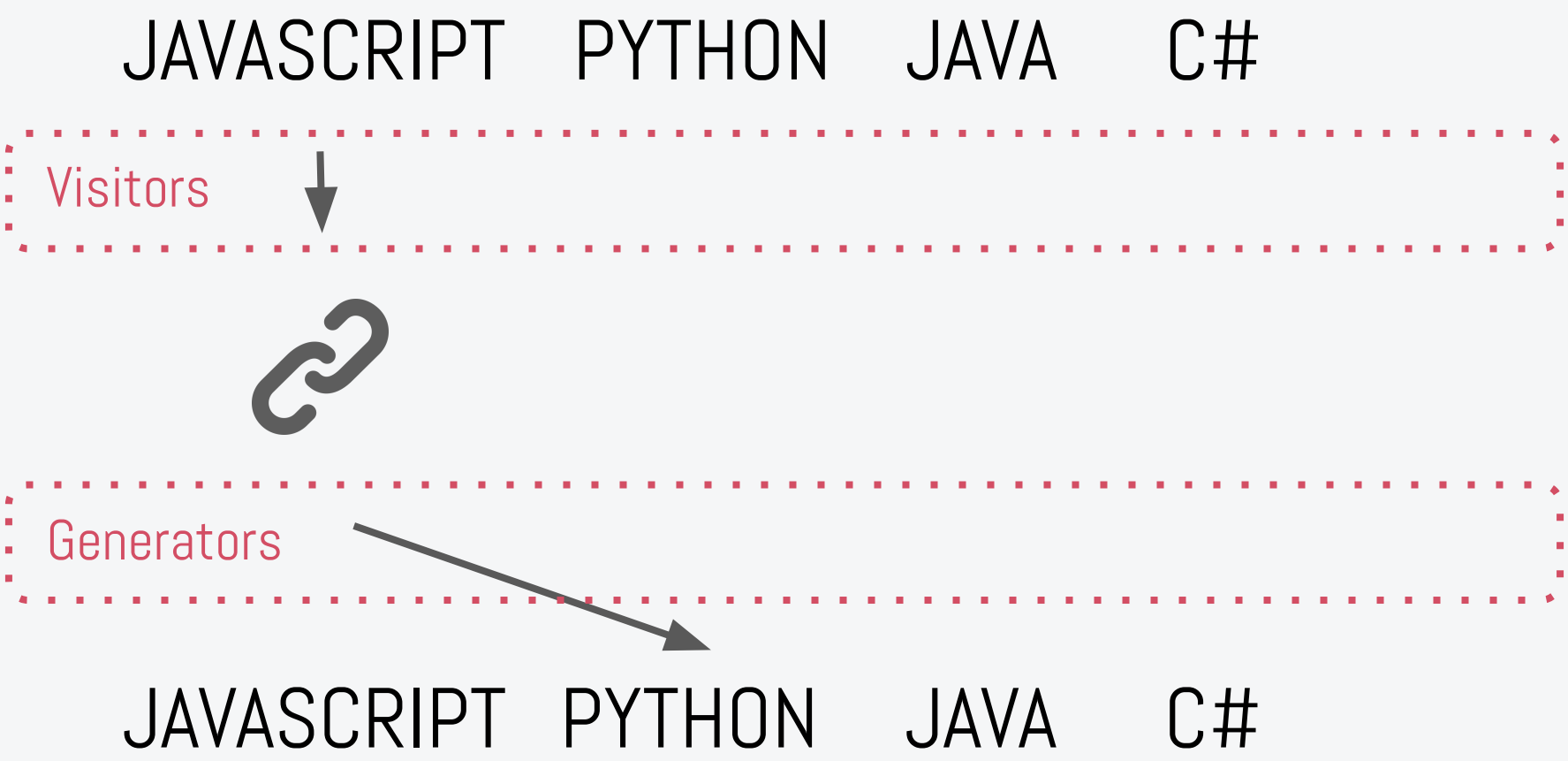
```
8
9     visitPropertyExpressionAssignment(ctx) {
10      const key = this.visit(ctx.propertyName());
11      const value = this.visit(ctx.singleExpression());
12
13      return `'${key}': ${value}`;
14    }
15
16    visitTerminal(ctx) {
17      return ctx.getText();
18    }
19
20    visitNumberExpression(ctx) {
21      const argumentList = ctx.arguments().argumentList();
22      const arg = argumentList.singleExpression()[0];
23      const number = this.visit(arg);
24
25      return `int(${number})`;
26    }
27  }
28
29  module.exports = Visitor;
30
```


JAVASCRIPT
PYTHON
Visitors
JAVASCRIPT
PYTHON
JAVA
C#

JS PythonGenerator.js ●

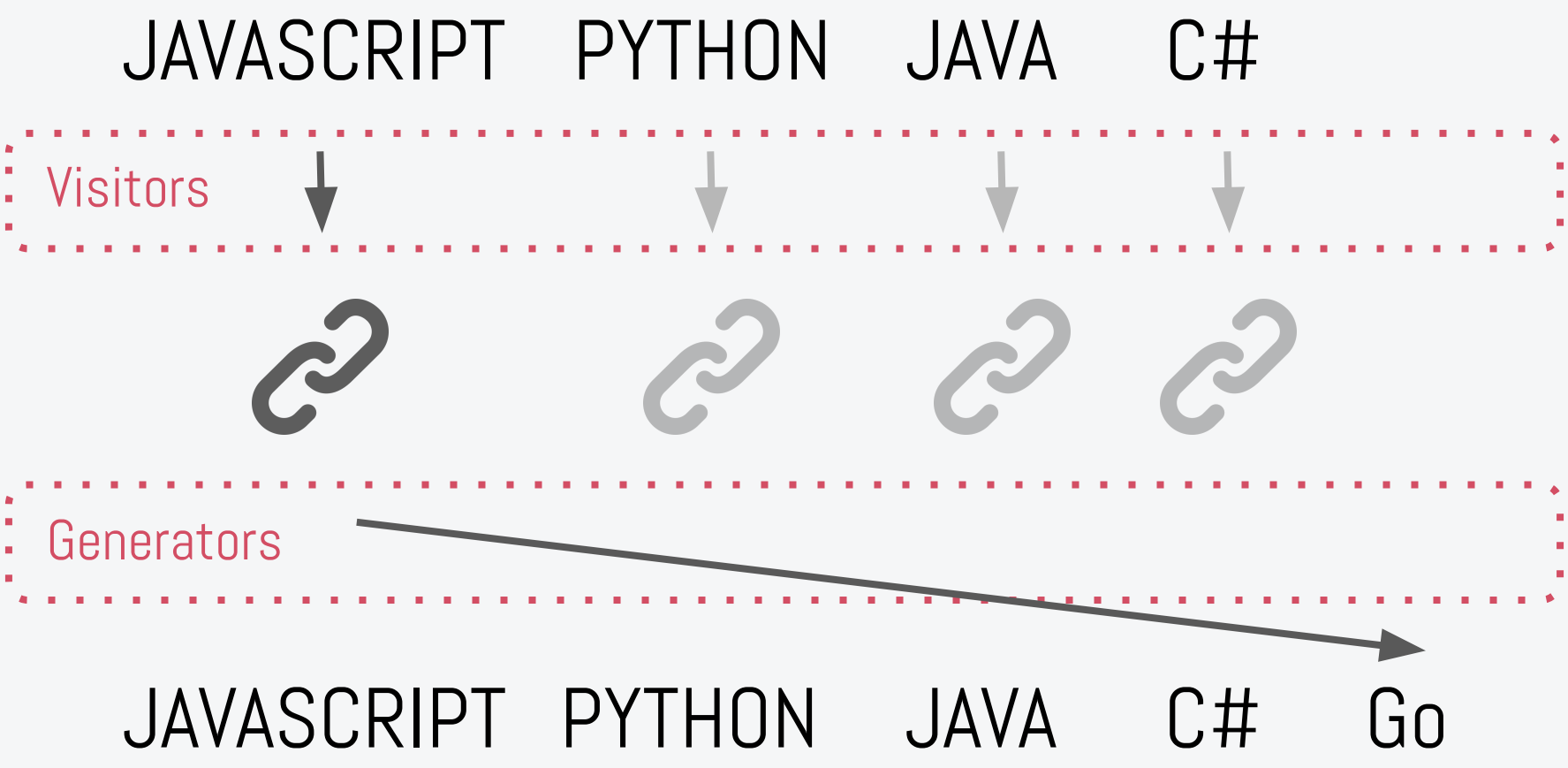
```
20    visitNumberExpression(ctx) {
21      const argumentList = ctx.arguments().argumentList();
22      const arg = argumentList.singleExpression()[0];
23      const number = this.visit(arg);
24
25      switch(outputLanguage) {
26        case 'Python':
27          return `int(${number})`;
28        case 'Java':
29          return '...';
30        case 'C#':
31          return '...';
32        default:
33          return `Number(${number})`;
34      }
35    }
36  }
37
38  module.exports = Visitor;
39
```


JAVASCRIPT PYTHON JAVA C#
Visitors
Generators
JAVASCRIPT PYTHON JAVA C#


JAVASCRIPT PYTHON JAVA C#
Visitors
Generators
JAVASCRIPT PYTHON JAVA C#

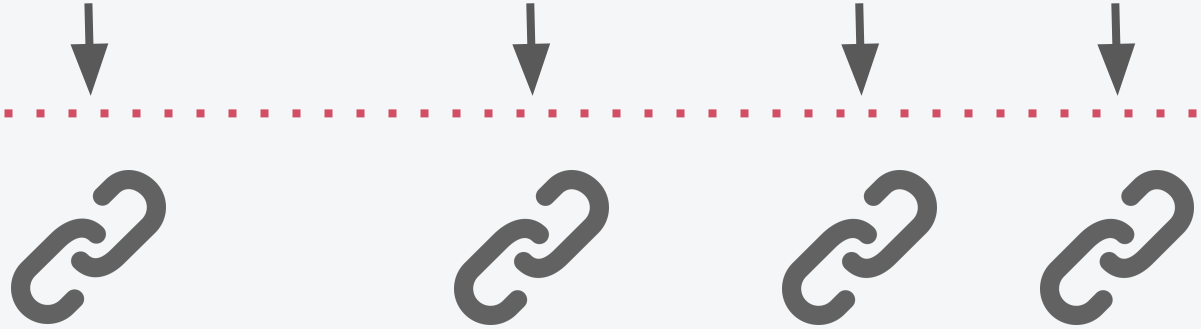
JS index.js

```
const getJavascriptVisitor = require('./codegeneration/javascript/Visitor');
const getPythonGenerator = require('./codegeneration/python/Generator');

// Lexing and parsing are happening somewhere here

composeTranspiler = (visitor, generator) => {
  const Transpiler = generator(visitor);
  const transpiler = new Transpiler();

  return {
    compile: (input) => {
      const tree = loadTree(input);
      return transpiler.start(tree);
    }
  };
}

module.exports = {
  javascript: {
    python: composeTranspiler(
      getJavascriptVisitor(JavascriptANTLRVisitor),
      getPythonGenerator
    )
  }
}
```

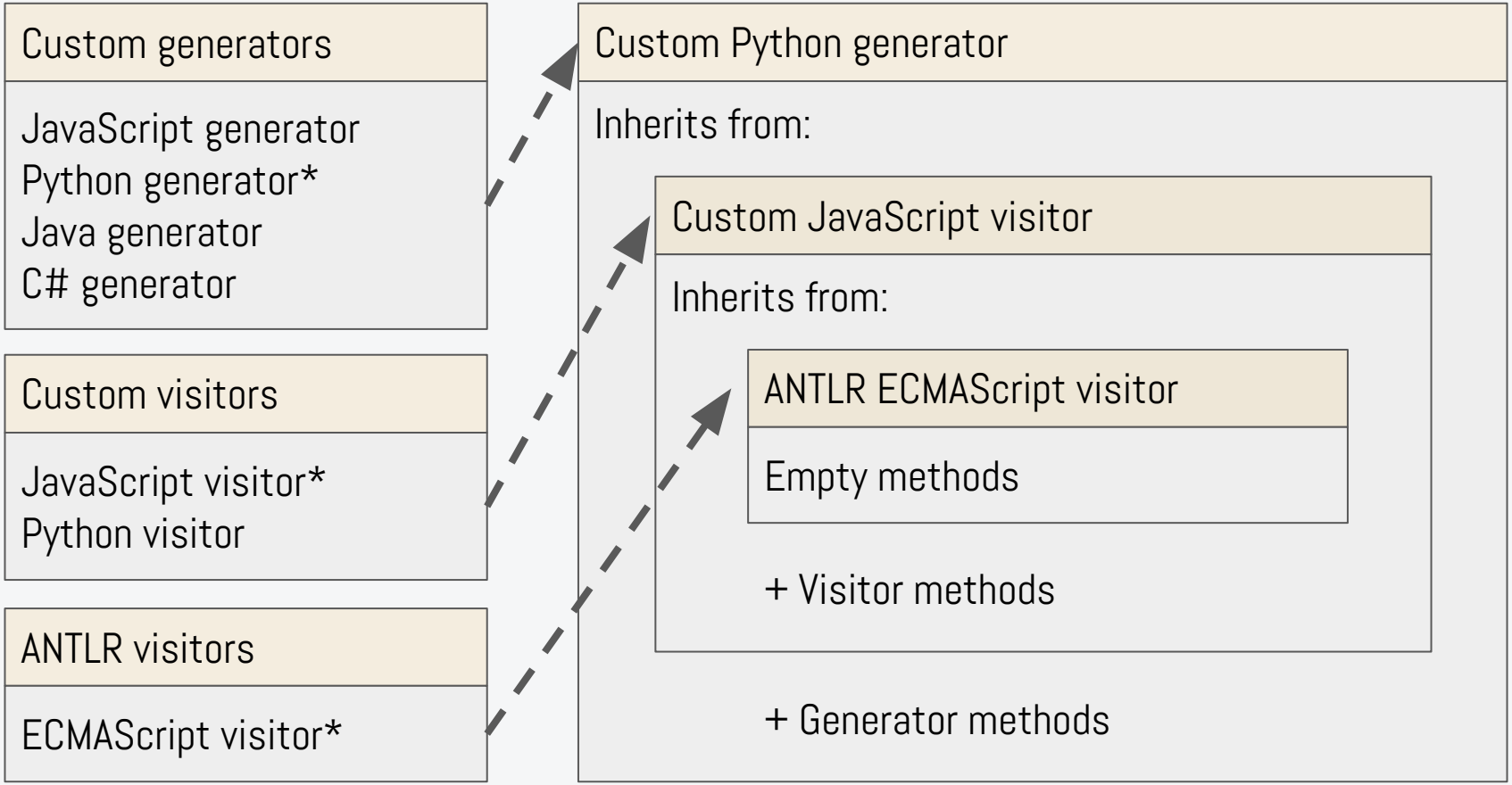

JAVASCRIPT PYTHON JAVA C#
Visitors
Generators
JAVASCRIPT PYTHON JAVA C# Go

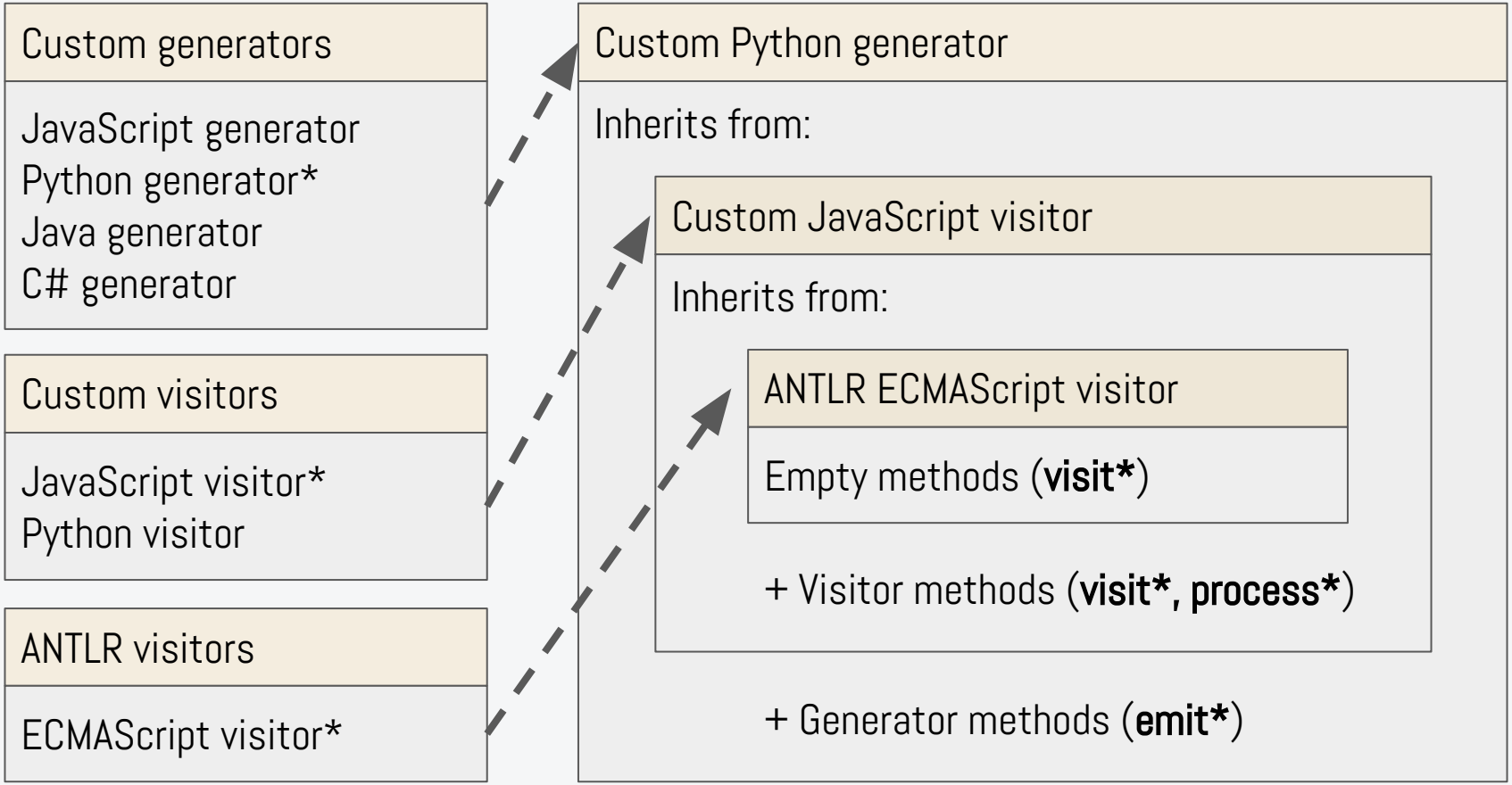
JAVASCRIPT PYTHON JAVA C#

Visitors



Generators


Custom generators
JavaScript generator
Python generator*
Java generator
C# generator
Custom visitors
JavaScript visitor*
Python visitor
ANTLR visitors
ECMAScript visitor*
Custom Python generator
Inherits from:
Custom JavaScript visitor
Inherits from:
ANTLR ECMAScript visitor
Empty methods
+ Visitor methods
+ Generator methods


Custom generators
JavaScript generator
Python generator*
Java generator
C# generator
Custom visitors
JavaScript visitor*
Python visitor
ANTLR visitors
ECMAScript visitor*
Custom Python generator
Inherits from:
Custom JavaScript visitor
Inherits from:
ANTLR ECMAScript visitor
Empty methods (visit*)
+ Visitor methods (visit*, process*)
+ Generator methods (emit*)

JS Visitor.js

```
1  const ECMAScriptVisitor = require('../lib/ECMAScriptVisitor').ECMAScriptVisitor;
2
3  class Visitor extends ECMAScriptVisitor {
4    start(ctx) {
5      return this.visitExpressionSequence(ctx);
6    }
7
8    visitFuncCallExpression(ctx) {
9      const lhs = this.visit(ctx.functionName());
10     const rhs = this.visit(ctx.arguments())
11
12     if (`process${lhs}` in this) {
13       return this[`process${lhs}`](ctx);
14     }
15
16     if (`emit${lhs}` in this) {
17       return this[`emit${lhs}`](ctx);
18     }
19
20     return `${lhs}${rhs}`;
21   }
22 }
23
24 module.exports = Visitor;
25
```

JS Visitor.js

```
1   const ECMAScriptVisitor = require('../lib/ECMAScriptVisitor').ECMAScriptVisitor;
2
3   class Visitor extends ECMAScriptVisitor {
4     start(ctx) {
5       return this.visitExpressionSequence(ctx);
6     }
7
8     visitFuncCallExpression(ctx) {
9       const lhs = this.visit(ctx.functionName());
10      const rhs = this.visit(ctx.arguments())
11
12      if (`process${lhs}` in this) {
13        return this[`process${lhs}`](ctx);
14      }
15
16      if (`emit${lhs}` in this) {
17        return this[`emit${lhs}`](ctx);
18      }
19
20      return `${lhs}${rhs}`;
21    }
22  }
23
24  module.exports = Visitor;
25
```

JS Visitor.js

```
const ECMAScriptVisitor = require('../lib/ECMAScriptVisitor').ECMAScriptVisitor;

class Visitor extends ECMAScriptVisitor {
  start(ctx) {
    return this.visitExpressionSequence(ctx);
  }

  visitFuncCallExpression(ctx) {
    const lhs = this.visit(ctx.functionName());
    const rhs = this.visit(ctx.arguments())

    if (`process${lhs}` in this) {
      return this[`process${lhs}`](ctx);
    }

    if (`emit${lhs}` in this) {
      return this[`emit${lhs}`](ctx);
    }

    return `${lhs}${rhs}`;
  }
}

module.exports = Visitor;
```

JS Visitor.js

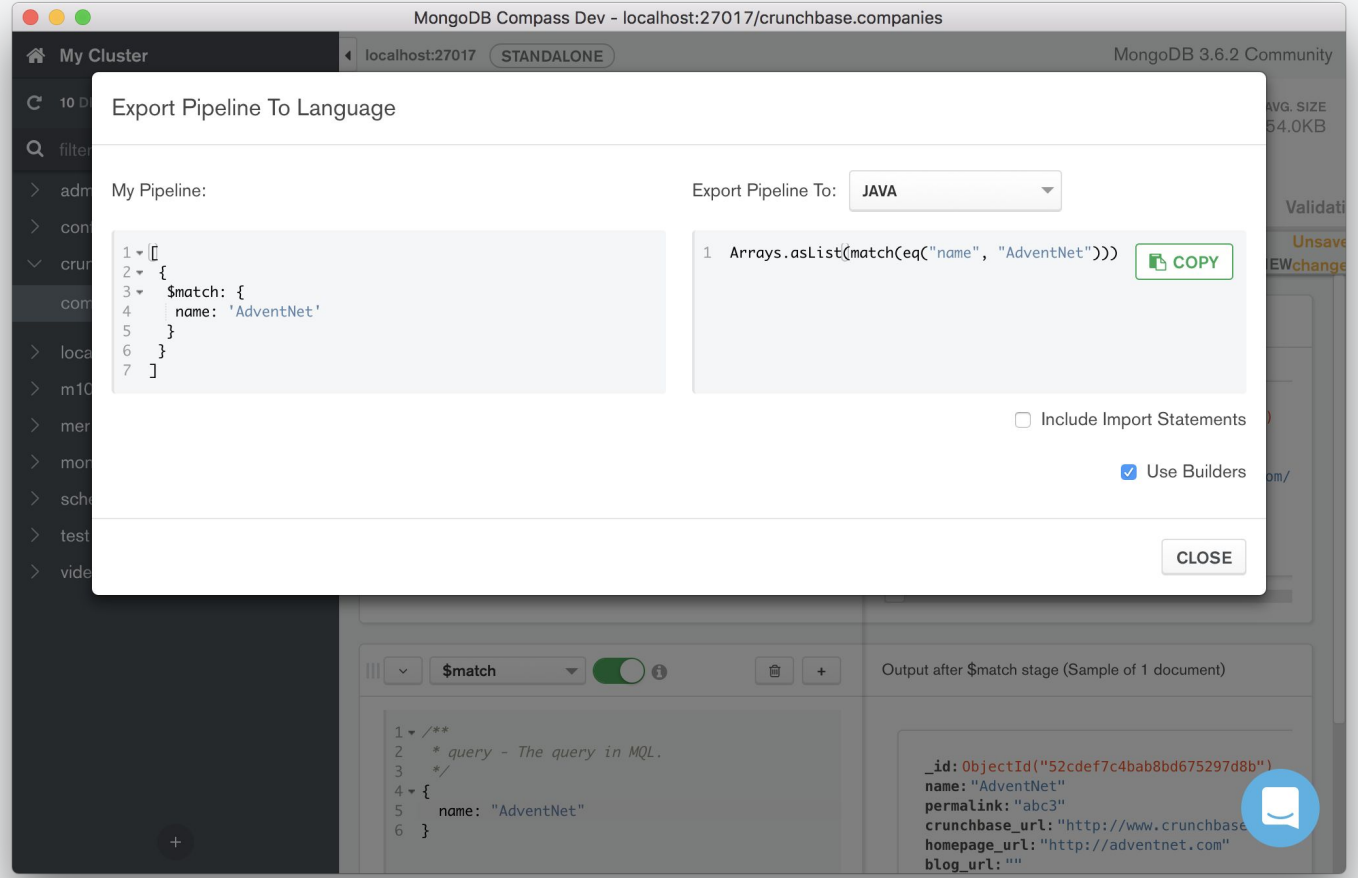
```
const ECMAScriptVisitor = require('../lib/ECMAScriptVisitor').ECMAScriptVisitor;

class Visitor extends ECMAScriptVisitor {
  start(ctx) {
    return this.visitExpressionSequence(ctx);
  }

  visitFuncCallExpression(ctx) {
    const lhs = this.visit(ctx.functionName());
    const rhs = this.visit(ctx.arguments())

    if (`process${lhs}` in this) {
      return this[`process${lhs}`](ctx);
    }

    if (`emit${lhs}` in this) {
      return this[`emit${lhs}`](ctx);
    }

    return `${lhs}${rhs}`;
  }
}

module.exports = Visitor;
```

JS Visitor.js

```
const ECMAScriptVisitor = require('../lib/ECMAScriptVisitor').ECMAScriptVisitor;

class Visitor extends ECMAScriptVisitor {
  start(ctx) {
    return this.visitExpressionSequence(ctx);
  }

  // visitFuncCallExpression() is somewhere here

  processDate(ctx) {
    let text = node.getText();
    let date;

    try {
      date = this.executeJavascript(text);
    } catch (error) {
      // Error
    }

    if ('emitDate' in this) {
      return this.emitDate(ctx, date);
    }
  }
}

module.exports = Visitor;
```

JS Generator.js

```
module.exports = (superClass) => class Generator extends superClass {
  emitDate(node, date) {
    const dateStr = [
      date.getUTCFullYear(),
      date.getUTCMonth() + 1,
      date.getUTCDate(),
      date.getUTCHours(),
      date.getUTCMinutes(),
      date.getUTCSeconds()
    ].join(', ');
    return `datetime.datetime(${dateStr}, tzinfo=datetime.timezone.utc)`;
  }
};

```


MongoDB Compass Dev - localhost:27017/crunchbase.companies
My Cluster
localhost:27017
STANDALONE
MongoDB 3.6.2 Community
Export Pipeline To Language
My Pipeline:
[
{
$match: {
name: 'AdventNet'
}
}
]
Export Pipeline To:
JAVA
Arrays.asList(match(eq("name", "AdventNet")))
COPY
Include Import Statements
Use Builders
CLOSE
$match
Output after $match stage (Sample of 1 document)
/**
* query - The query in MQL.
*/
{
name: "AdventNet"
}
_id: ObjectId("52cdef7c4bab8bd675297d8b")
name: "AdventNet"
permalink: "abc3"
homepage_url: "http://adventnet.com"
blog_url: ""

SUMMARY

SUMMARY

- Applied general compiler design principles

SUMMARY

- Applied general compiler design principles
- Used ANTLR to build a parse tree

SUMMARY

- Applied general compiler design principles
- Used ANTLR to build a parse tree
- Introduced visitors and generators

SUMMARY

- Applied general compiler design principles
- Used ANTLR to build a parse tree
- Introduced visitors and generators
- Added optimisation by using string templates

JS Generator.js

```
1  module.exports = (Visitor) => class Generator extends Visitor {
2    constructor() {
3      super();
4    }
5  };
6
```

! templates.yaml

```
267    LongToIntTemplate: &LongToIntTemplate !!js/function >
268        (lhs) => {
269            return `int(${lhs})`;
270        }
271    LongToIntArgsTemplate: &LongToIntArgsTemplate !!js/function >
272        () => {
273            return '';
274        }
275    LongToStringTemplate: &LongToStringTemplate !!js/function >
276        () => {
277            return 'str';
278        }
```

Contributing to bson-transpilers

Setting up your environment

The `bson-transpilers` package uses ANTLR4 to create a parse tree. As `ANTLR` is written in Java, you will need to set up a few tools before being able to compile this locally.

Make sure you have Java installed:

```
$ brew cask install java
```

Download `ANTLR4`:

```
$ cd /usr/local/lib && curl -O http://www.antlr.org/download/antlr-4.7.1-complete.jar
```

You will then need to add it to your `$CLASSPATH`:

```
$ export CLASSPATH=".:/usr/local/lib/antlr-4.7.1-complete.jar:$CLASSPATH"
```

ALENA KHINEIKA

 @alena_khineika

 @alena_khineika

 @alenakhineika