创心 第十四届 D2前端技术论坛

2019 - 12 - 14 杭州和达希尔顿逸林酒店

创心 第十四届 D2前端技术论坛

翻译 / Translation

BABEL

Under the Hood

NICOLÒ RIBAUDO

Babel team

@NicoloRibaudo

@nicolo-ribaudo

@nicolo-ribaudo

nicolo.ribaudo@gmail.com

What is Babel?

What is Babel?

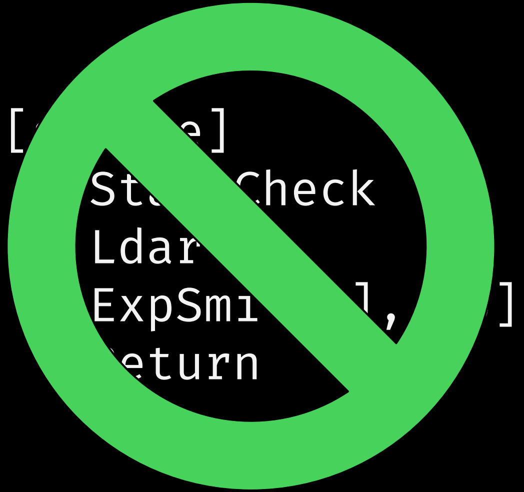
Babel is a JavaScript compiler

```js
const square = n => n ** 2;
```

```
[square]
    StackCheck
    Ldar a0
    ExpSmi [2], [0]
    Return
```

@NicoloRibaudo

```
const square = n => n ** 2;
```



```
[     e]
  St    Check
  Ldar
  ExpSmi      ],    ]
  eturn
```

It’s a JavaScript to JavaScript compiler

```
const square = n => n ** 2;
```

```
var square = function (n) {
 return Math.pow(n, 2);
};
```

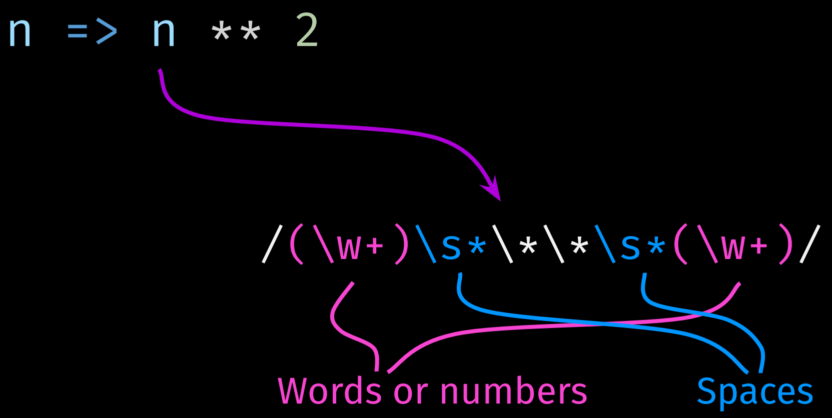
Compilers’ data structure:

Abstract Syntax Tree (AST)

A naive approach to compilation: *Regular Expressions*

```
n => n ** 2
```

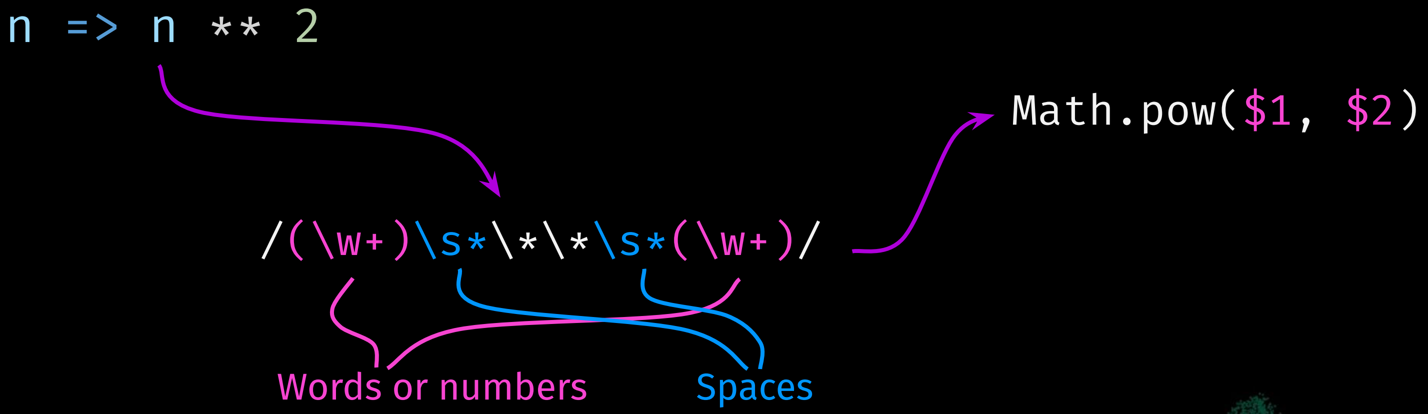
A naive approach to compilation: *Regular Expressions*



```
n => n ** 2
```

```
/(\w+)\s*\*\*\s*(\w+)/
```

A naive approach to compilation: *Regular Expressions*

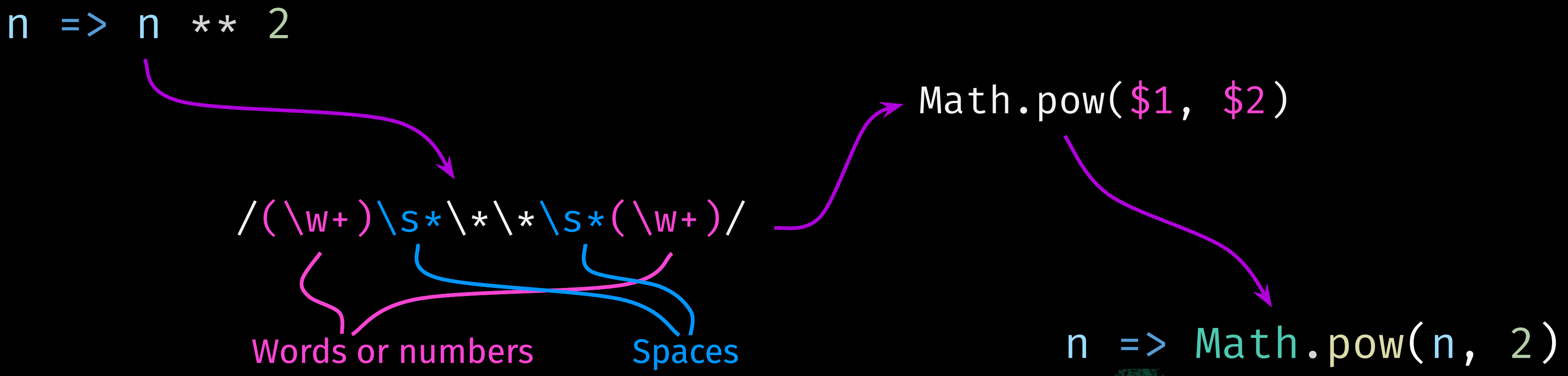


```
n => n ** 2
```

```
/(\w+)\s*\*\*\s*(\w+)/
```

```
Math.pow($1, $2)
```

A naive approach to compilation: *Regular Expressions*



```
n => n ** 2
```

```
/(\w+)\s*\*\*\s*(\w+)/
```

```
Math.pow($1, $2)
```

```
n => Math.pow(n, 2)
```

@NicoloRibaudo

A naive approach to compilation: *Regular Expressions*

🙁

@NicoloRibaudo

A naive approach to compilation: *Regular Expressions*

☹️

COMPLEXITY

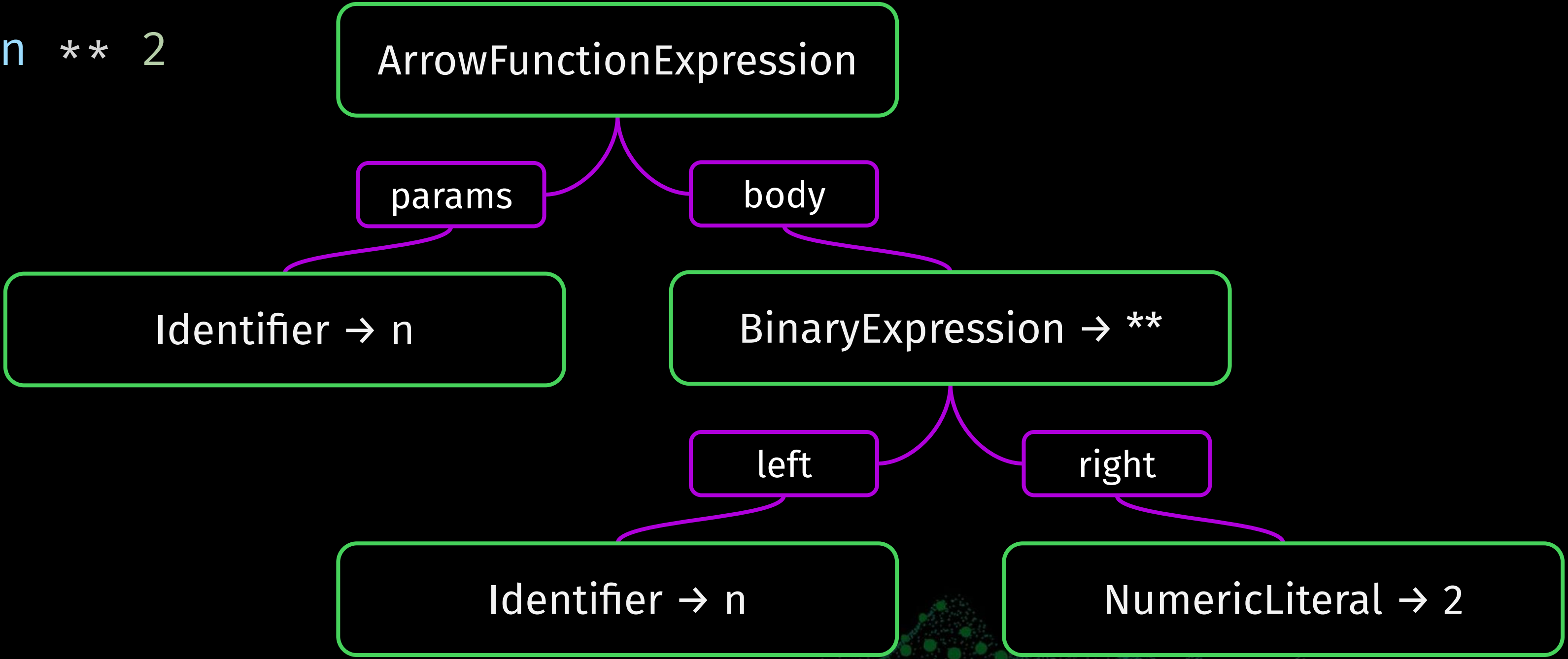
`fn(a) ** (2 ** 3)`

INACCURACY

`"8" !== "2 ** 3"`

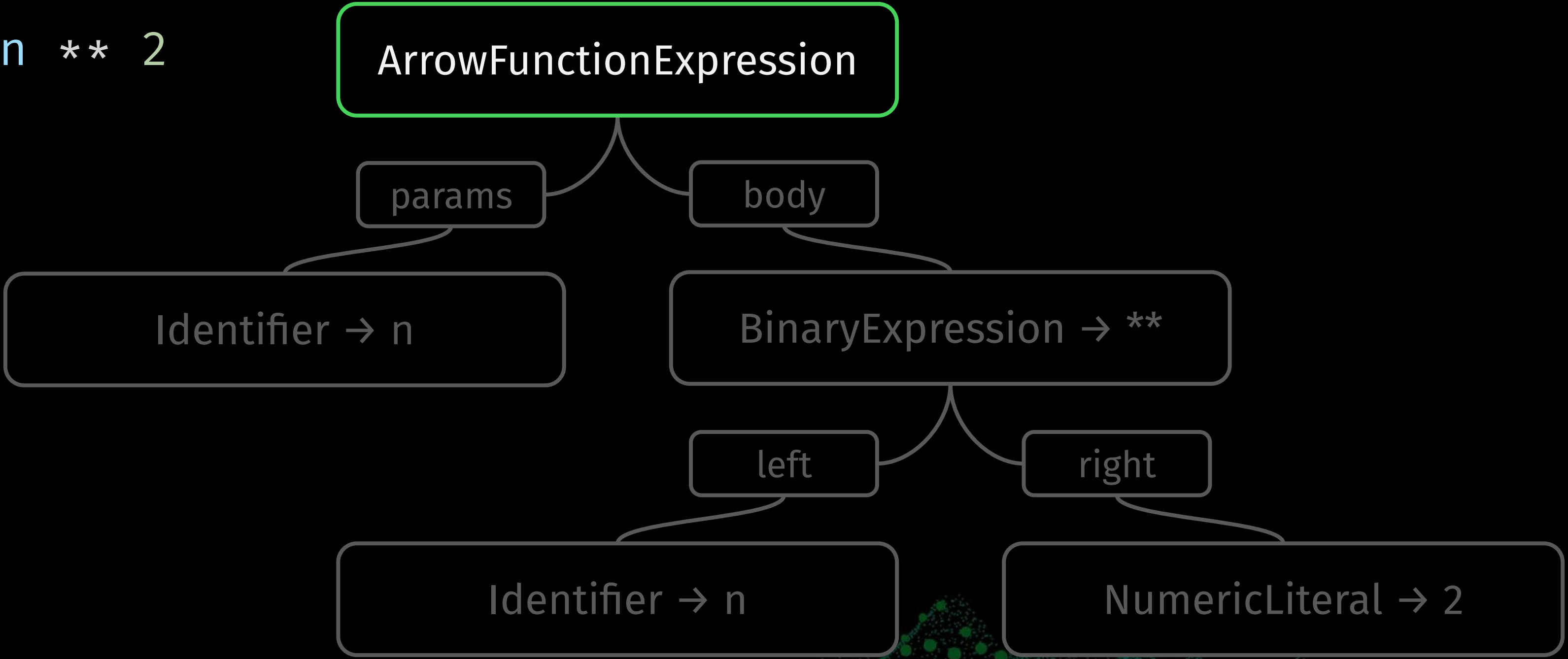
Abstract Syntax Tree

```
n => n ** 2
```



Abstract Syntax Tree

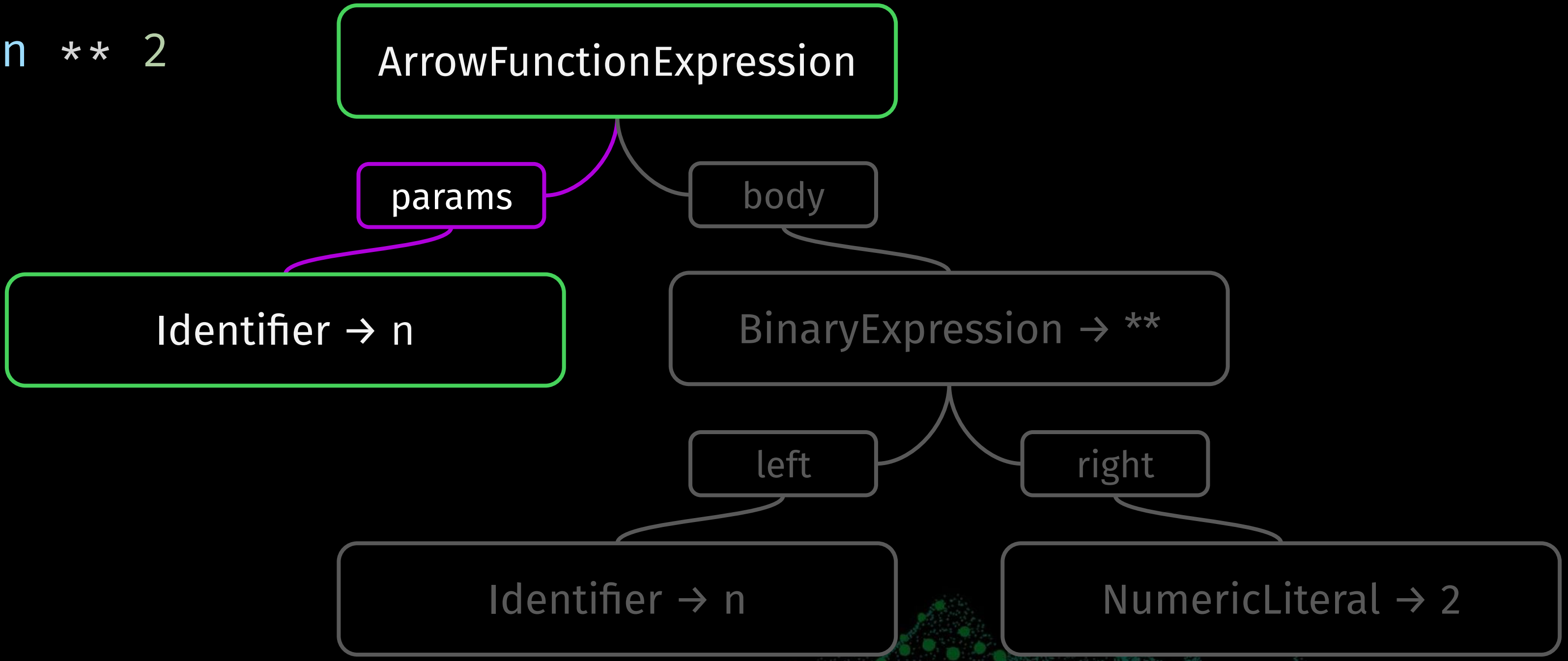
```
n => n ** 2
```



@NicoloRibaudo

Abstract Syntax Tree

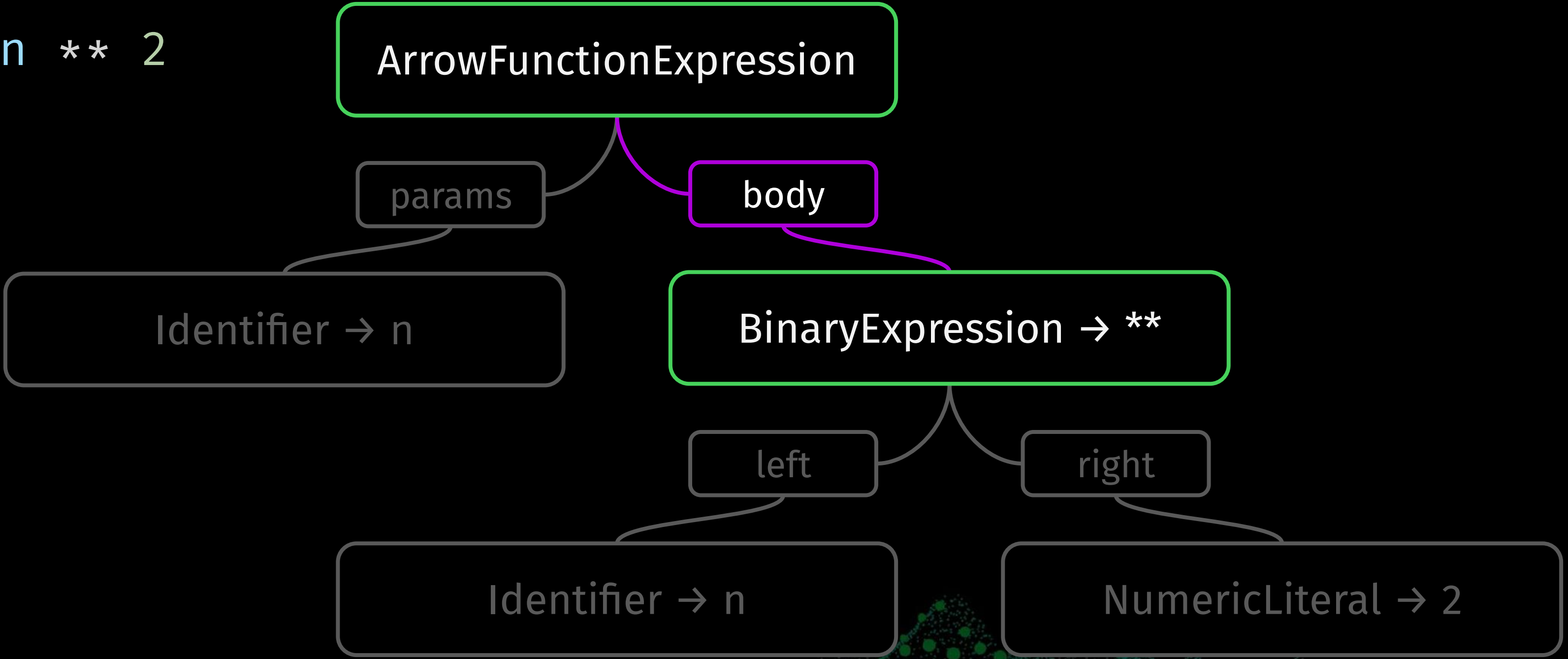
```
n => n ** 2
```



创心 第十四届 D2前端技术论坛

Abstract Syntax Tree

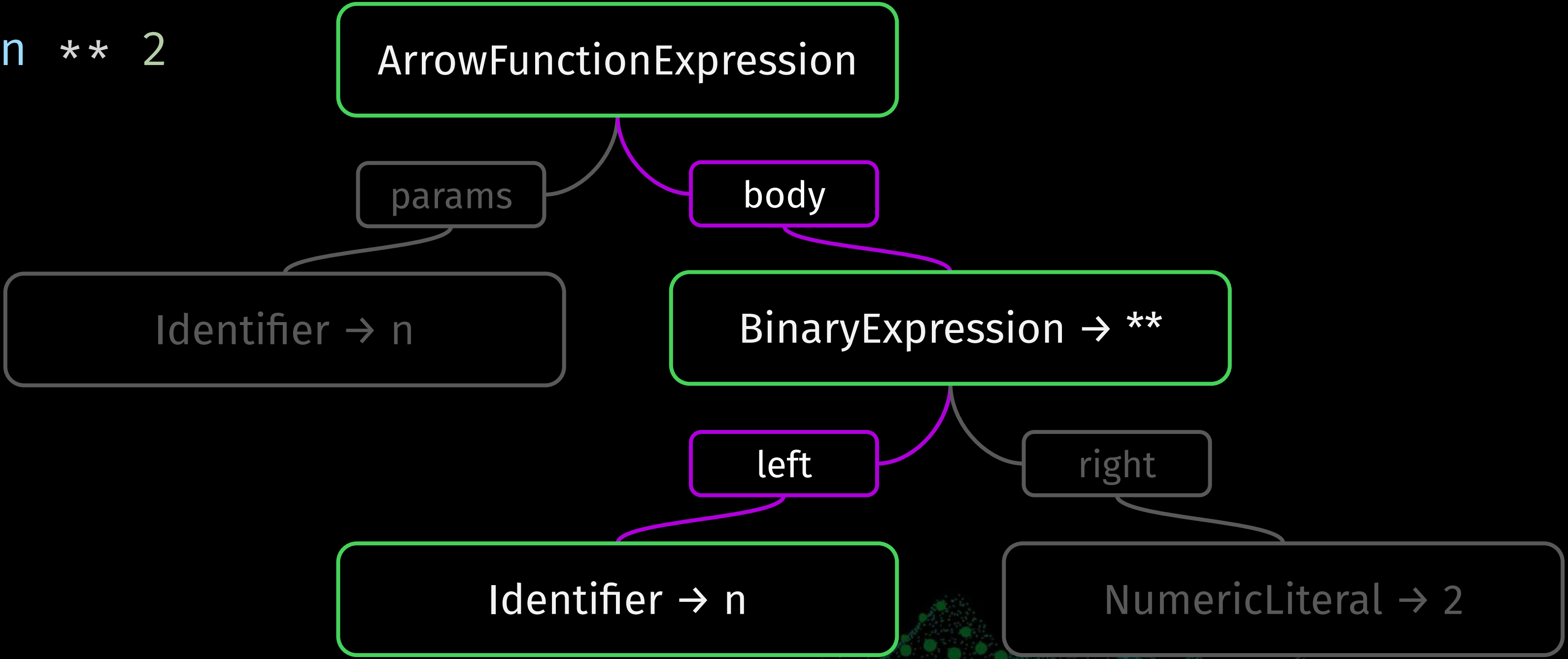
```
n => n ** 2
```



@NicoloRibaudo

Abstract Syntax Tree

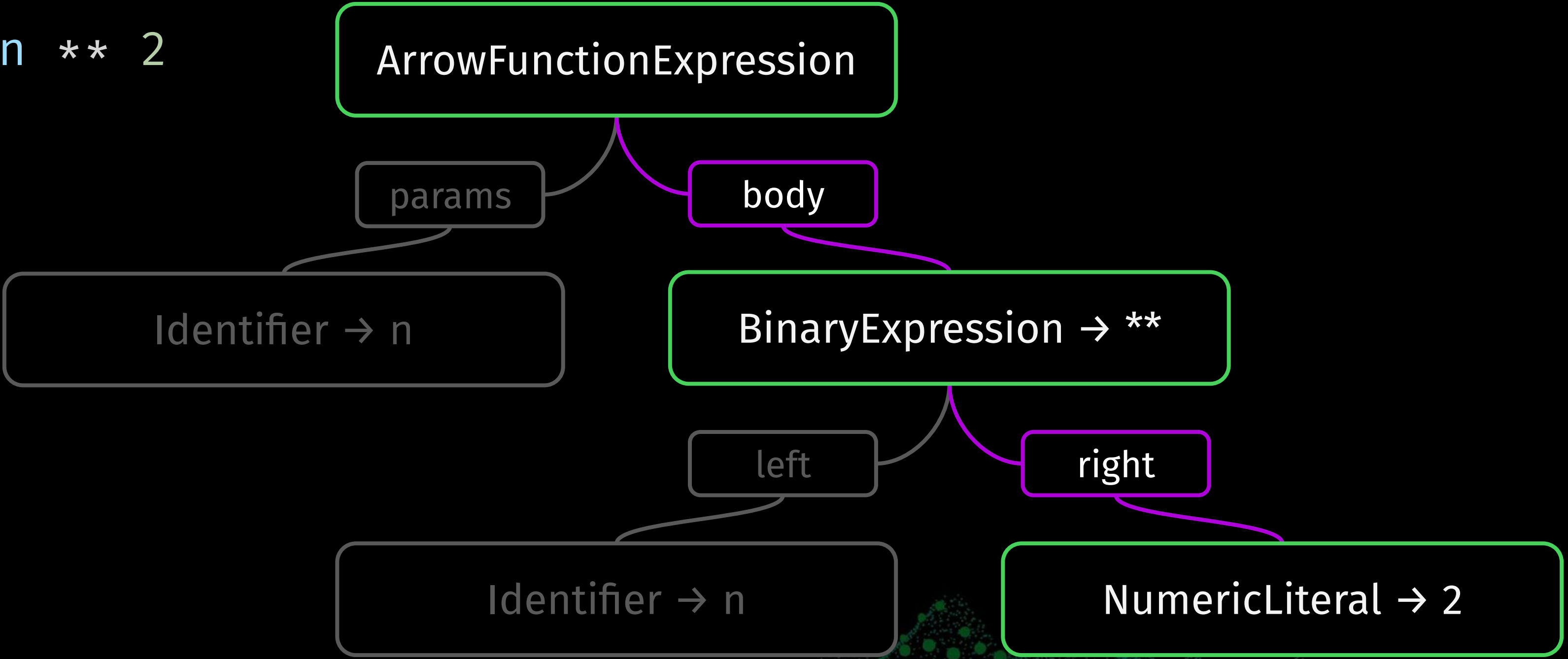
```
n => n ** 2
```



@NicoloRibaudo

Abstract Syntax Tree

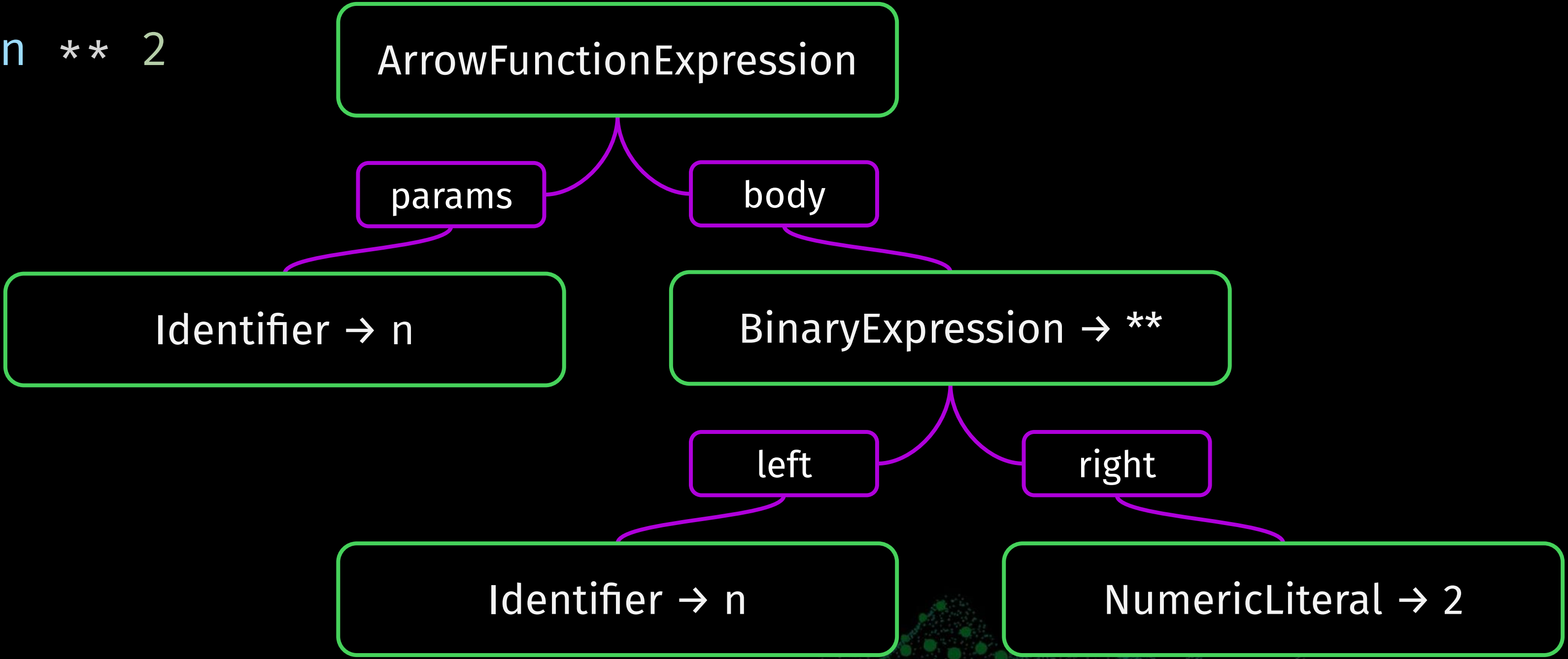
```
n => n ** 2
```



@NicoloRibaudo

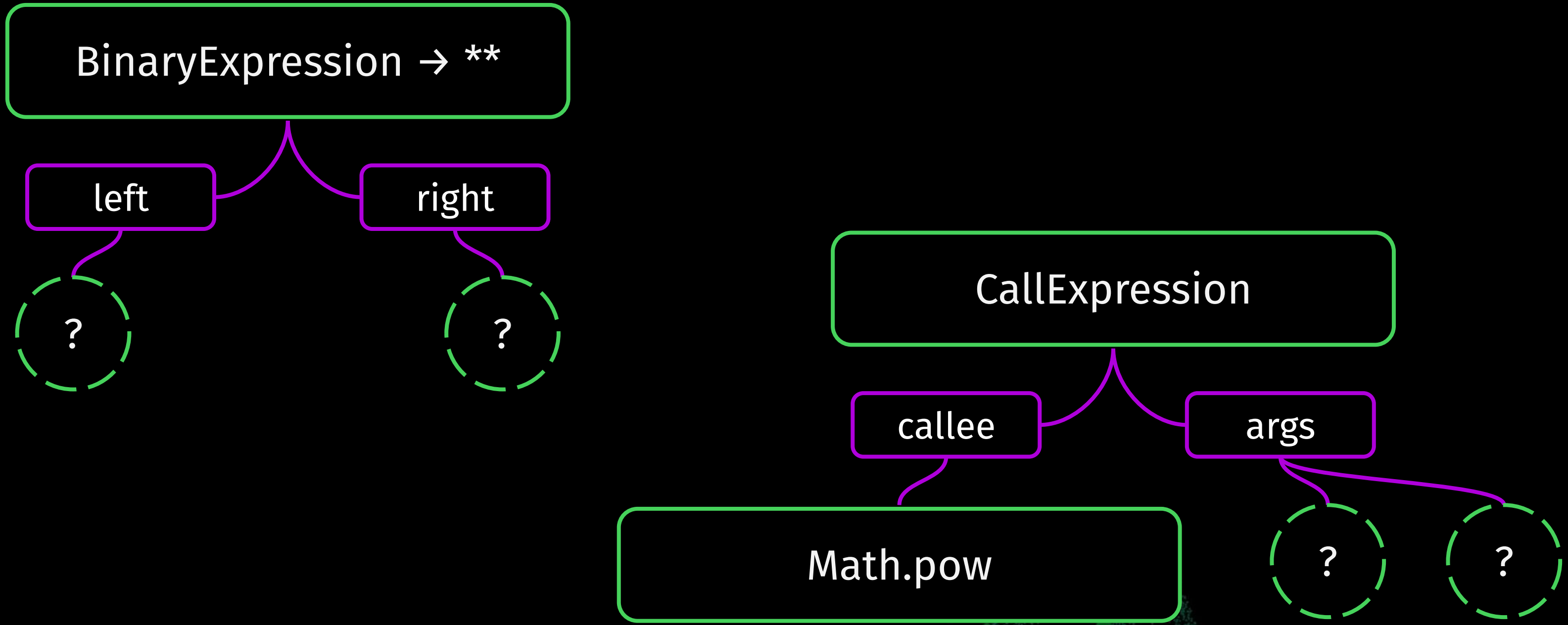
Abstract Syntax Tree

```
n => n ** 2
```



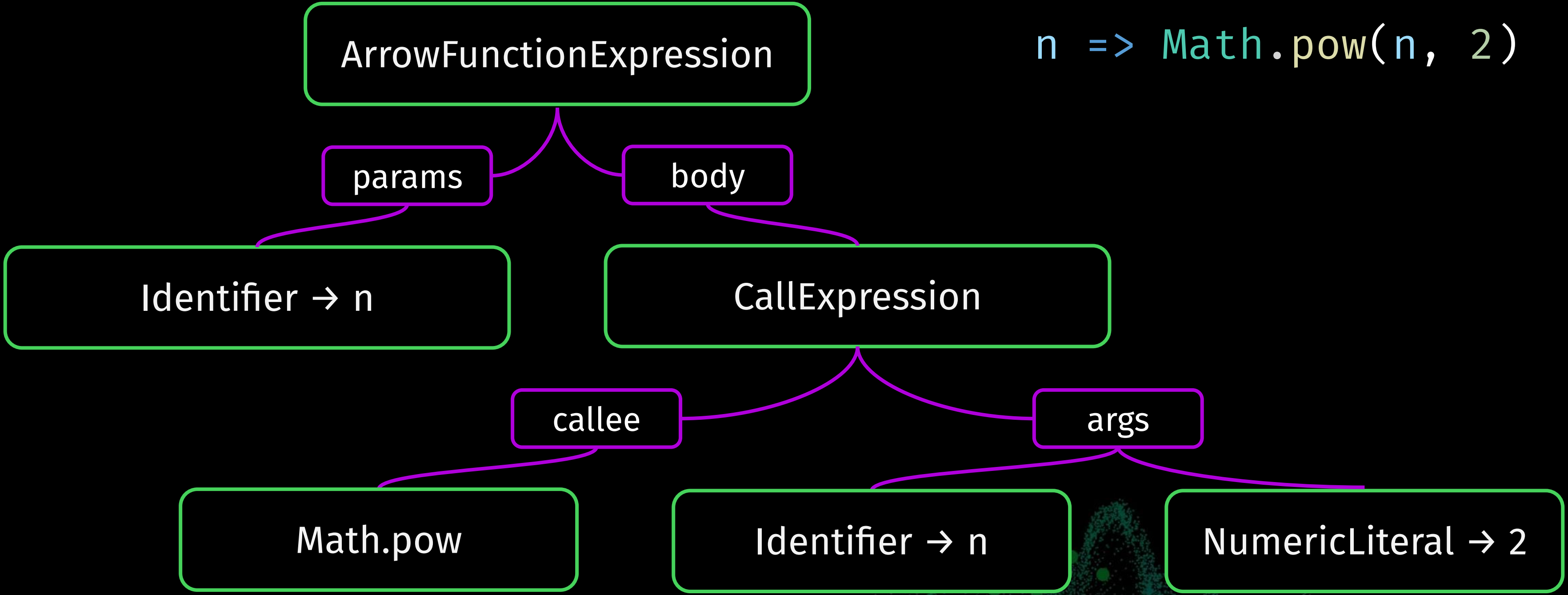
@NicoloRibaudo

AST node replacement



@NicoloRibaudo

AST node replacement



```
n => Math.pow(n, 2)
```

@NicoloRibaudo

AST node replacement

😄

AST node replacement

😄

~~COMPLEXITY~~

```
fn(a) ** (2 ** 3)
```

? ** ?

~~INACCURACY~~

```
"8" !== "2 ** 3"
```

StringLiteral → "2 ** 3"

Babel's AST

```
{
 "type": "BinaryExpression",
 "operator": "**",
 "left": {
   "type": "Identifier",
   "name": "n"
 },
 "right": {
   "type": "NumericLiteral",
   "value": 2
 }
}
```

```
{
 "type": "CallExpression",
 "callee": { /* ... */ },
 "arguments": [{
   "type": "Identifier",
   "name": "n"
 }, {
   "type": "NumericLiteral",
   "value": 2
 }]
}
```

Babel's AST

```
{
 "type": "BinaryExpression",
 "operator": "**",
 "left": {
   "type": "Identifier",
   "name": "n"
 },
 "right": {
   "type": "NumericLiteral",
   "value": 2
 }
}
```

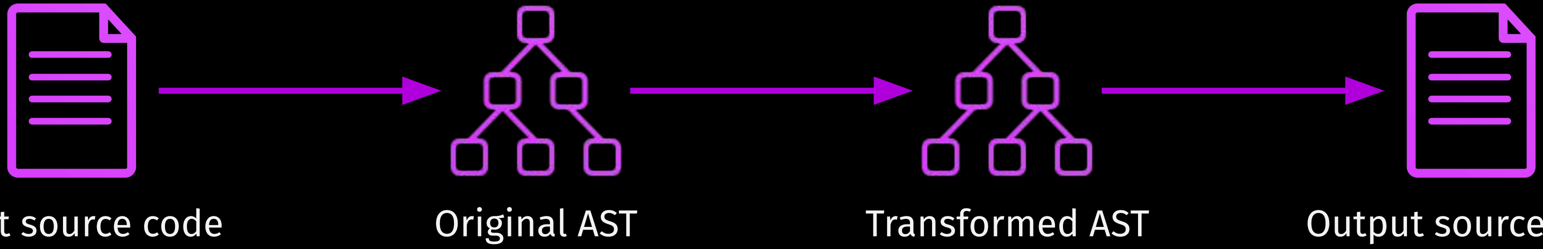
```
{
 "type": "CallExpression",
 "callee": { /* ... */ },
 "arguments": [{
   "type": "Identifier",
   "name": "n"
 }, {
   "type": "NumericLiteral",
   "value": 2
 }]
}
```

@NicoloRibaudo

A look inside Babel

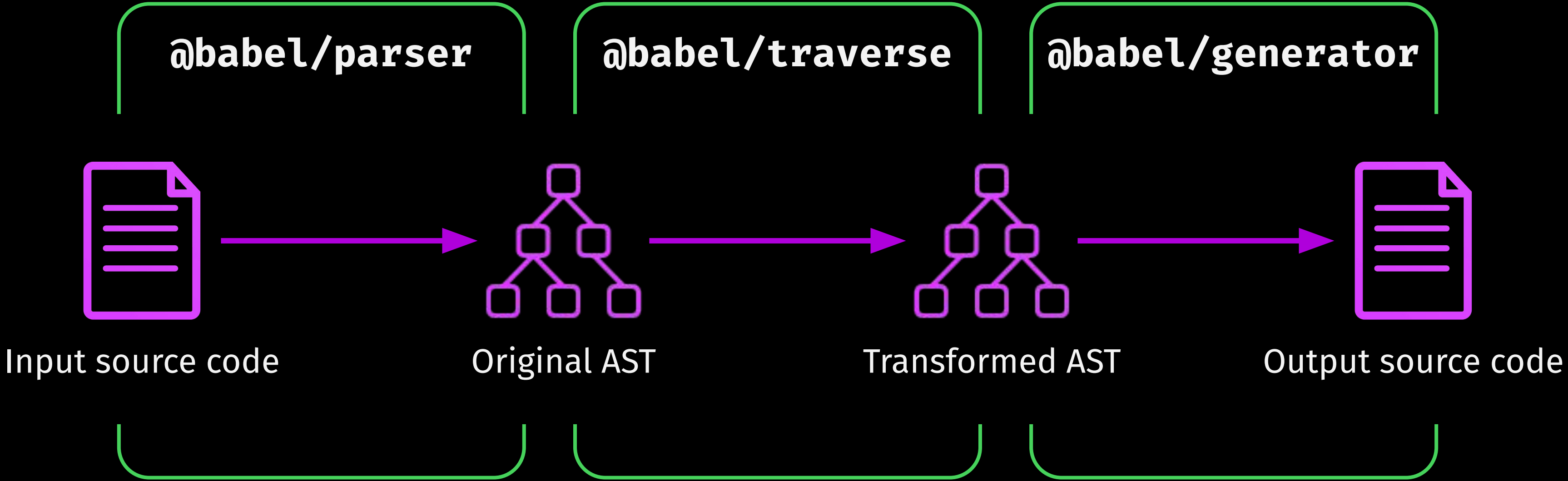
创心 第十四届 D2前端技术论坛

A look inside Babel

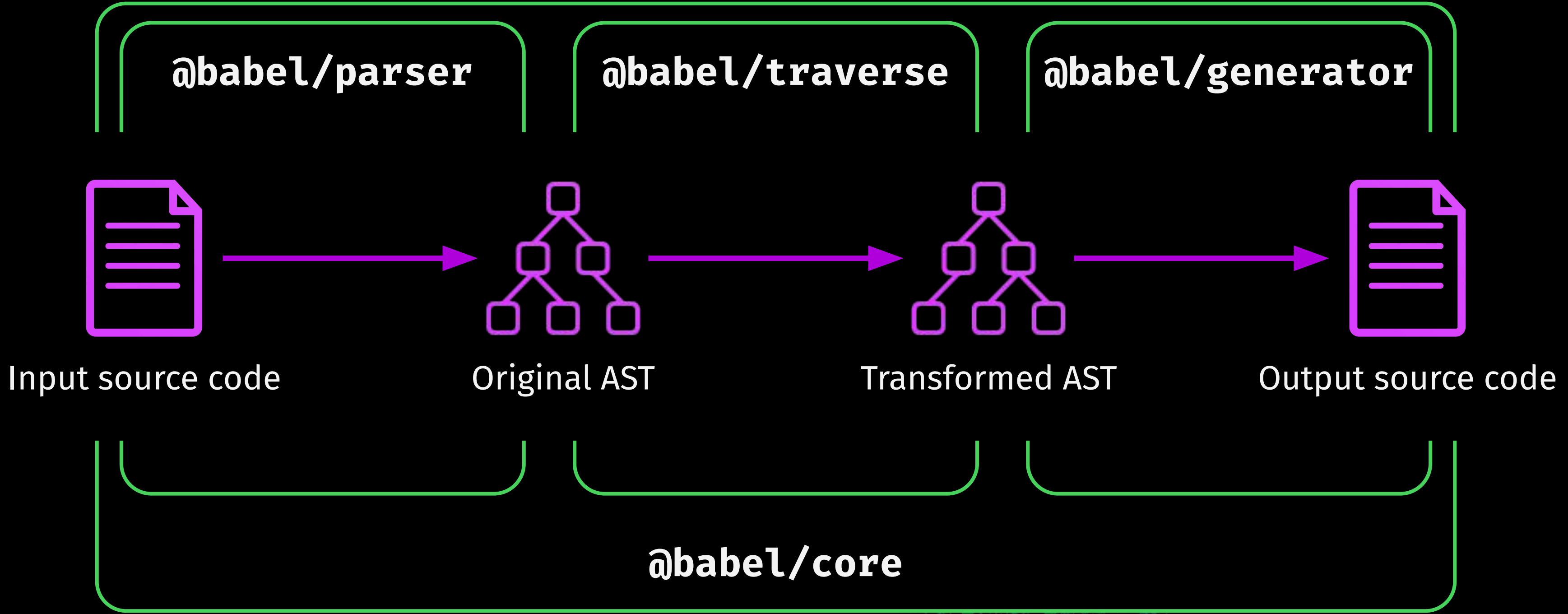


@NicoloRibaudo

A look inside Babel

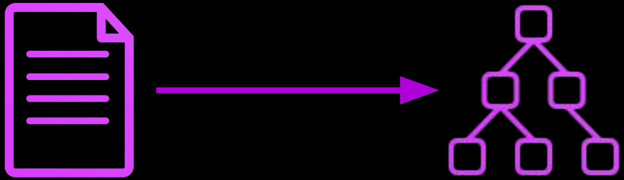


A look inside Babel



A look inside Babel:

`@babel/parser`



@NicoloRibaudo

1. Lexical analysis

Transform the input source code into a list of ***tokens***

```
var a = 7;
```

创心 第十四届 D2前端技术论坛

1. Lexical analysis

Transform the input source code into a list of ***tokens***

```
var a = 7;
```

1. Keyword	`var`

@NicoloRibaudo

1. Lexical analysis

Transform the input source code into a list of ***tokens***

```
var a = 7;
```

1. Keyword	`var`
2. Identifier	`a`

1. Lexical analysis

Transform the input source code into a list of ***tokens***

```
var a = 7;
```

1.	Keyword	`var`
2.	Identifier	`a`
3.	Punctuator	`=`

1. Lexical analysis

Transform the input source code into a list of ***tokens***

```
var a = 7;
```

1.	Keyword	`var`
2.	Identifier	`a`
3.	Punctuator	`=`
4.	Literal	`7`

1. Lexical analysis

Transform the input source code into a list of ***tokens***

```
var a = 7;
```

1.	Keyword	`var`
2.	Identifier	`a`
3.	Punctuator	`=`
4.	Literal	`7`
5.	Punctuator	`;`

1. Lexical analysis

Report errors about invalid literals or characters

1. Lexical analysis

Report errors about invalid literals or characters

Unterminated comment

```
/* var a = 7;
```

1. Lexical analysis

Report errors about invalid literals or characters

```
Unterminated comment              /* var a = 7;

Unexpected character '°'          var a = 7°;
```

1. Lexical analysis

Report errors about invalid literals or characters

Unterminated comment	`/* var a = 7;`
Unexpected character '°'	`var a = 7°;`
Expected number in radix 2	`var a = 0b20;`

2. Syntax analysis

Transform the list of tokens into an ***AST***

```
var a = 7;
```

2. Syntax analysis

Transform the list of tokens into an ***AST***

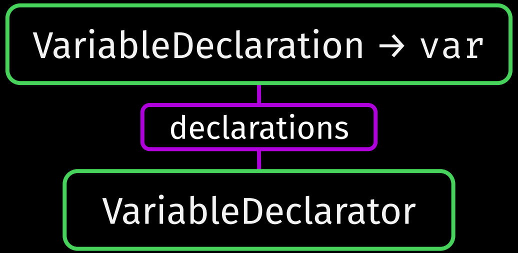
```
var a = 7;
```

VariableDeclaration → `var`

2. Syntax analysis

Transform the list of tokens into an ***AST***

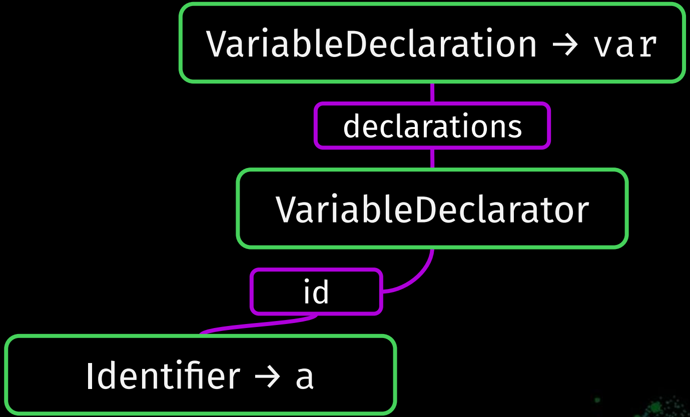
```
var a = 7;
```



2. Syntax analysis

Transform the list of tokens into an ***AST***

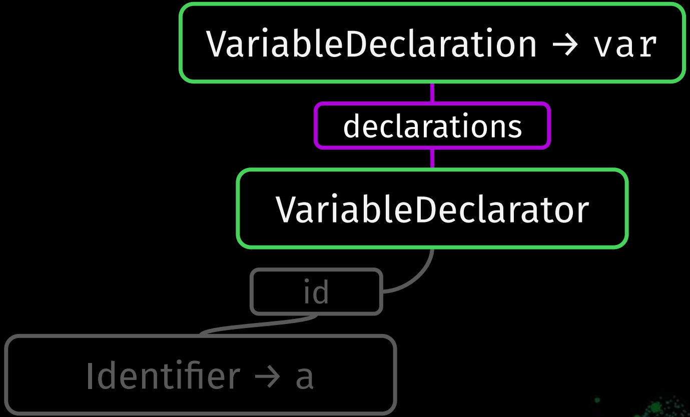
```
var a = 7;
```



2. Syntax analysis

Transform the list of tokens into an ***AST***

```
var a = 7;
```

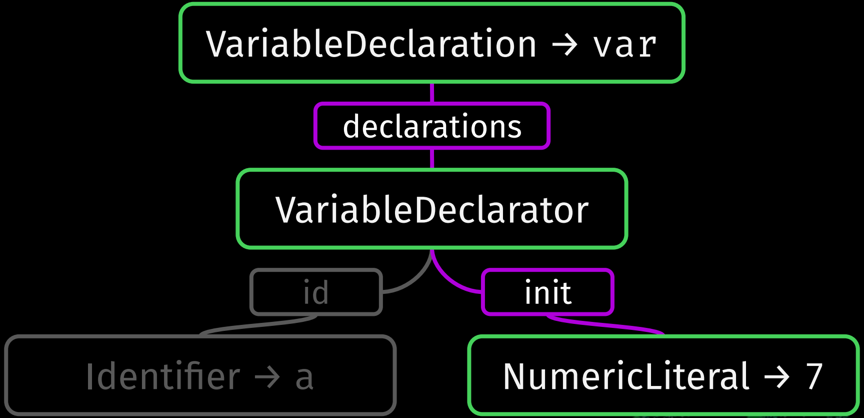


@NicoloRibaudo

2. Syntax analysis

Transform the list of tokens into an ***AST***

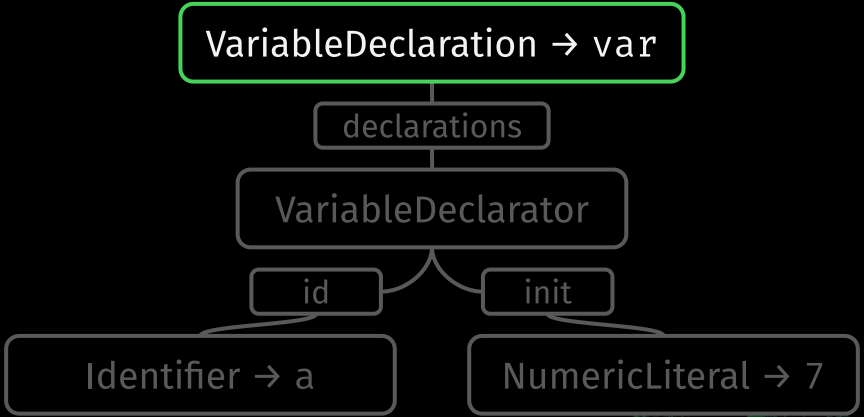
```
var a = 7;
```



2. Syntax analysis

Transform the list of tokens into an ***AST***

```
var a = 7;
```

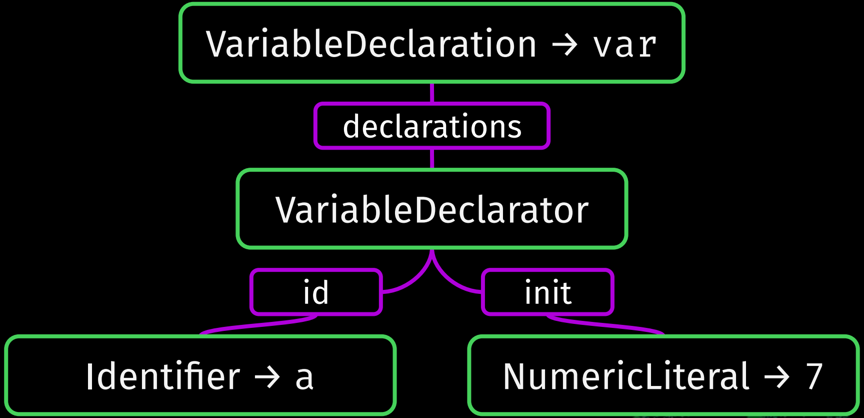


@NicoloRibaudo

2. Syntax analysis

Transform the list of tokens into an ***AST***

```
var a = 7;
```



2. Syntax analysis

Handle automatic semicolon insertion (*ASI*)

2. Syntax analysis

Handle automatic semicolon insertion (*ASI*)

```
var a = foo
foo.forEach(fn)
```

2. Syntax analysis

Handle automatic semicolon insertion (*ASI*)

```
var a = foo
foo.forEach(fn)
```

```
var a = foo;
foo.forEach(fn);
```

@NicoloRibaudo

2. Syntax analysis

Handle automatic semicolon insertion (*ASI*)

```
var a = foo
foo.forEach(fn)
```

```
var a = foo;
foo.forEach(fn);
```

```
var a = foo
[7].forEach(fn)
```

2. Syntax analysis

Handle automatic semicolon insertion (*ASI*)

```
var a = foo
foo.forEach(fn)
```

```
var a = foo;
foo.forEach(fn);
```

```
var a = foo
[7].forEach(fn)
```

```
var a = foo[7].forEach(n);
```

2. Syntax analysis

Report errors about misplaced tokens

2. Syntax analysis

Report errors about misplaced tokens

Unexpected token, expected ")"

```
var a = double(7;
```

2. Syntax analysis

Report errors about misplaced tokens

Unexpected token, expected ")"	`var a = double(7;`
Unexpected keyword 'if'	`1 + if;`

3. Semantic analysis

Check that the AST respects all the static ECMAScript rules: ***early errors***

3. Semantic analysis

Check that the AST respects all the static ECMAScript rules: ***early errors***

```
Redefinition of __proto__
property
```

```
({ __proto__: x,
   __proto__: y,
})
```

3. Semantic analysis

Check that the AST respects all the static ECMAScript rules: ***early errors***

Redefinition of __proto__ property

```
({ __proto__: x,
   __proto__: y,
})
```

'with' in strict mode

```
"use strict";
with (obj) {}
```

3. Semantic analysis

Report errors about invalid variables, using a ***scope tracker***

3. Semantic analysis

Report errors about invalid variables, using a ***scope tracker***

```
Identifier 'foo' has
already been declared
```

```
let foo = 2;
let foo = 3;
```

3. Semantic analysis

Report errors about invalid variables, using a ***scope tracker***

Identifier 'foo' has already been declared

```
let foo = 2;
let foo = 3;
```

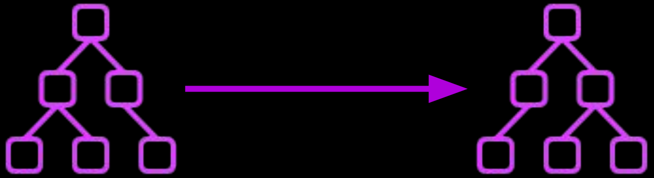
Export 'bar' is not defined

```
{ let bar = 2; }
export { bar };
```

@NicoloRibaudo

A look inside Babel:

`@babel/traverse`



Declarative traversal

Algorithm: Depth-first search, in-order (`enter`) and out-order (`exit`)

```
traverse(ast, {
  CallExpression: {
    enter() {
      console.log("Function call!")
    }
  }
})
```

Declarative traversal

Algorithm: Depth-first search, in-order (`enter`) and out-order (`exit`)

```
traverse(ast, {
  CallExpression: {
    enter() {
      console.log("Function call!")
    }
  }
})
```

Declarative traversal

Algorithm: Depth-first search, in-order (`enter`) and out-order (`exit`)

```
traverse(ast, {
  CallExpression: {
    enter() {
      console.log("Function call!")
    }
  }
})
```

@NicoloRibaudo

Declarative traversal

Algorithm: Depth-first search, in-order (`enter`) and out-order (`exit`)

```
traverse(ast, {
  CallExpression: {
    enter() {
      console.log("Function call!")
    }
  }
})
```

`enter` is the default traversal order

@NicoloRibaudo

Declarative traversal

Algorithm: Depth-first search, in-order (`enter`) and out-order (`exit`)

```
traverse(ast, {
  CallExpression() {

      console.log("Function call!")


  }
})
```

`enter` is the default traversal order

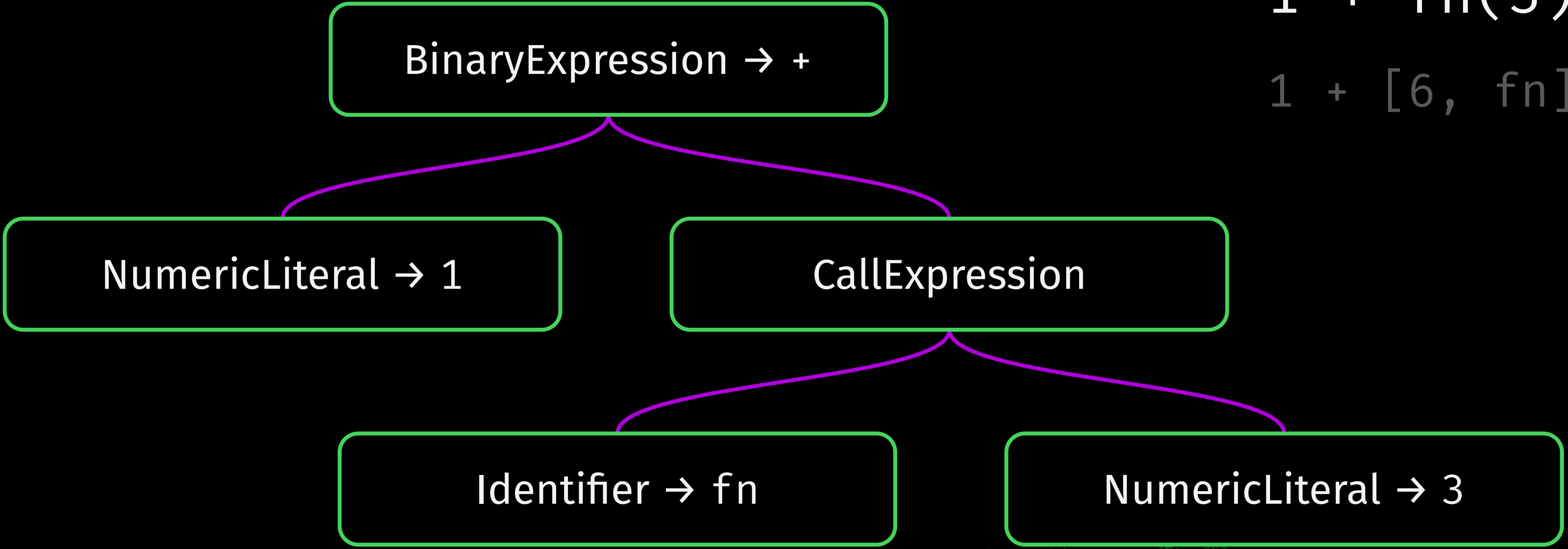
Dynamic Abstract Syntax Tree

```
1 + fn(3)
1 + [6, fn]
```

Example:

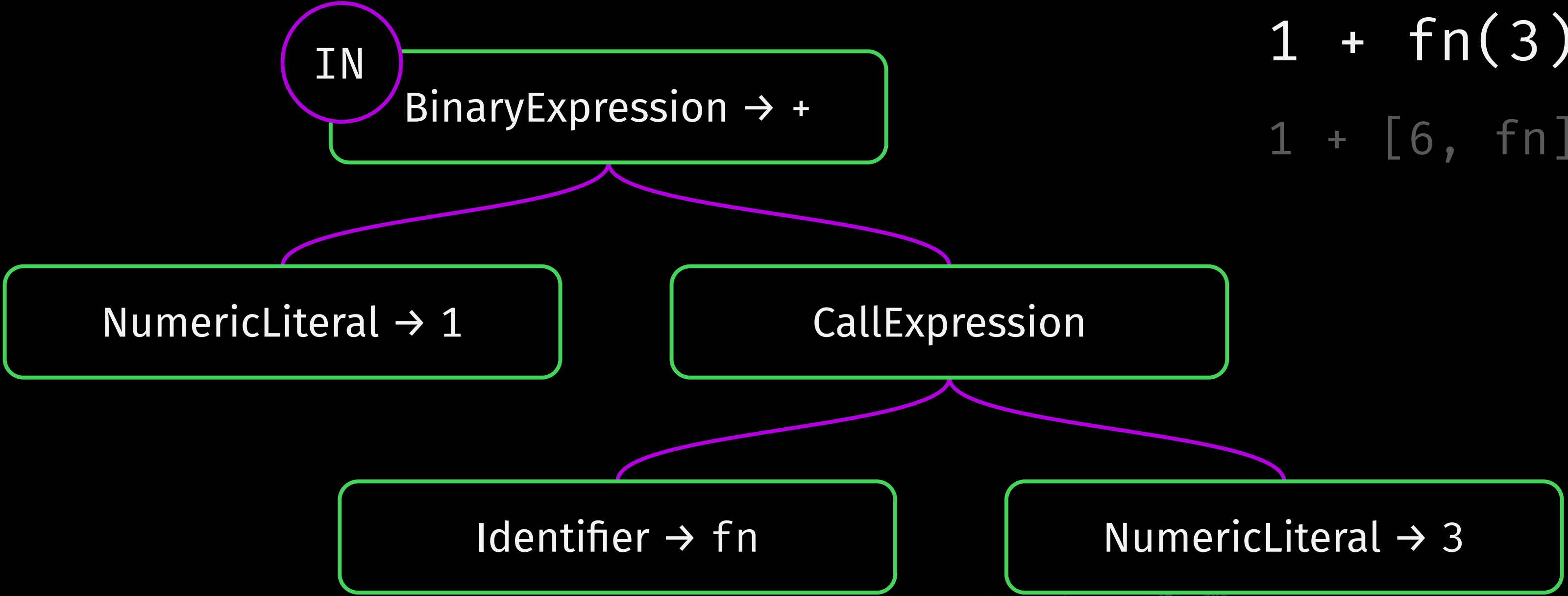
- ➔ Traverse `1 + fn(3)`
- ➔ When we reach `fn(3)`, during the "`exit`" phase, replace it with `[6, fn]`

Dynamic Abstract Syntax Tree



```
1 + fn(3)

1 + [6, fn]
```

Dynamic Abstract Syntax Tree

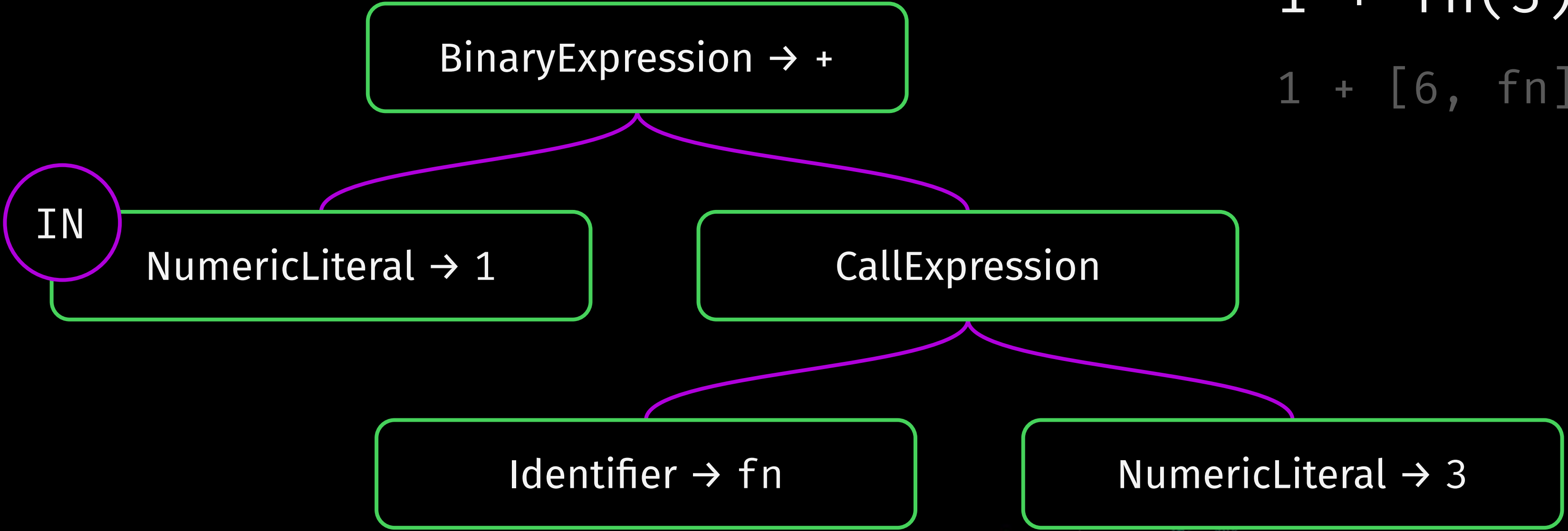


```
1 + fn(3)
1 + [6, fn]
```

@NicoloRibaudo

Dynamic Abstract Syntax Tree

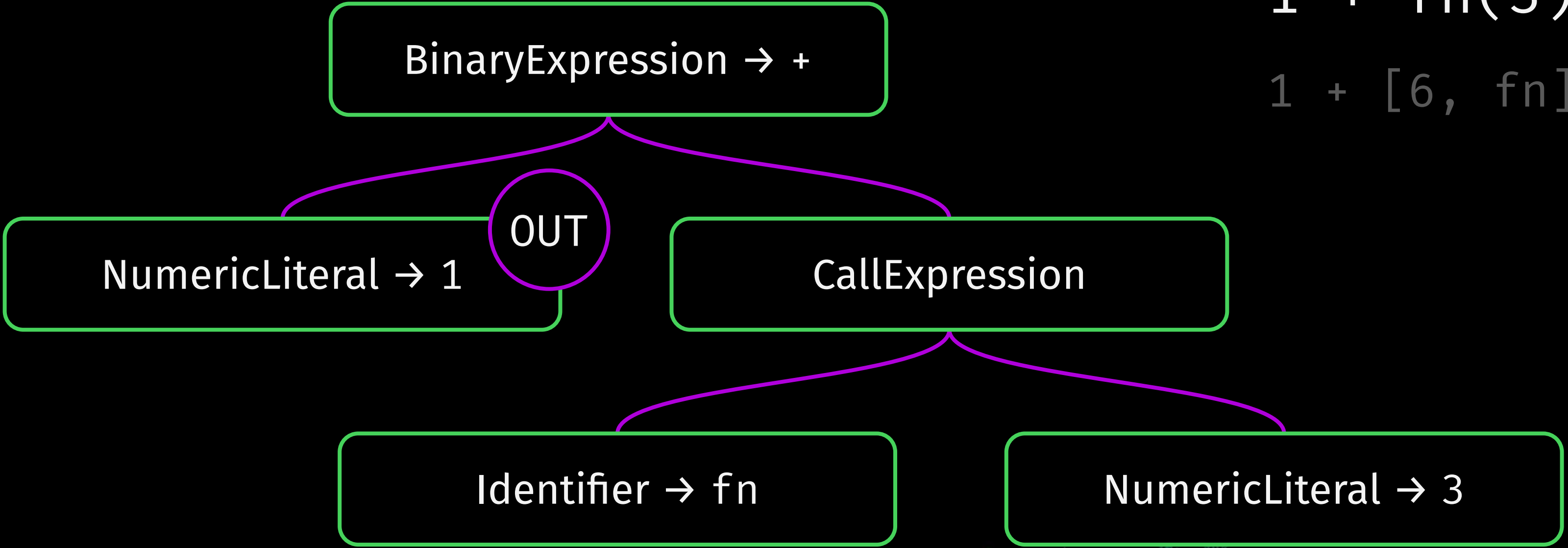
```
1 + fn(3)
1 + [6, fn]
```



Dynamic Abstract Syntax Tree

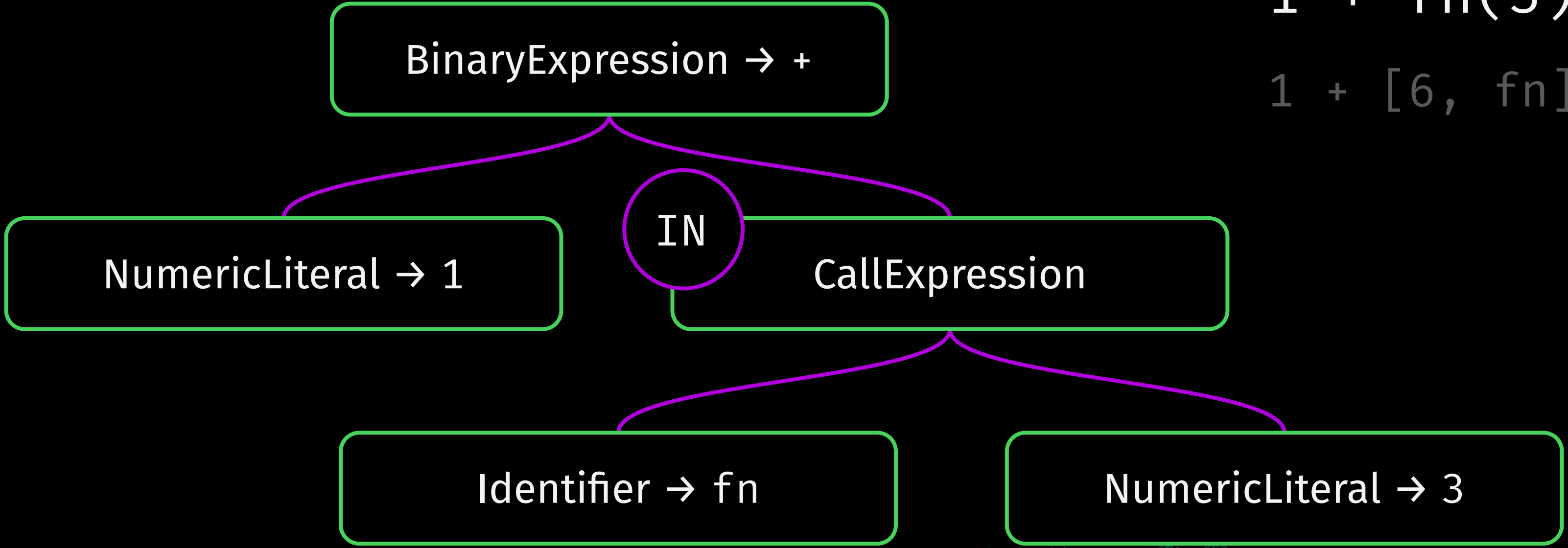
```
1 + fn(3)
```

```
1 + [6, fn]
```



@NicoloRibaudo

Dynamic Abstract Syntax Tree

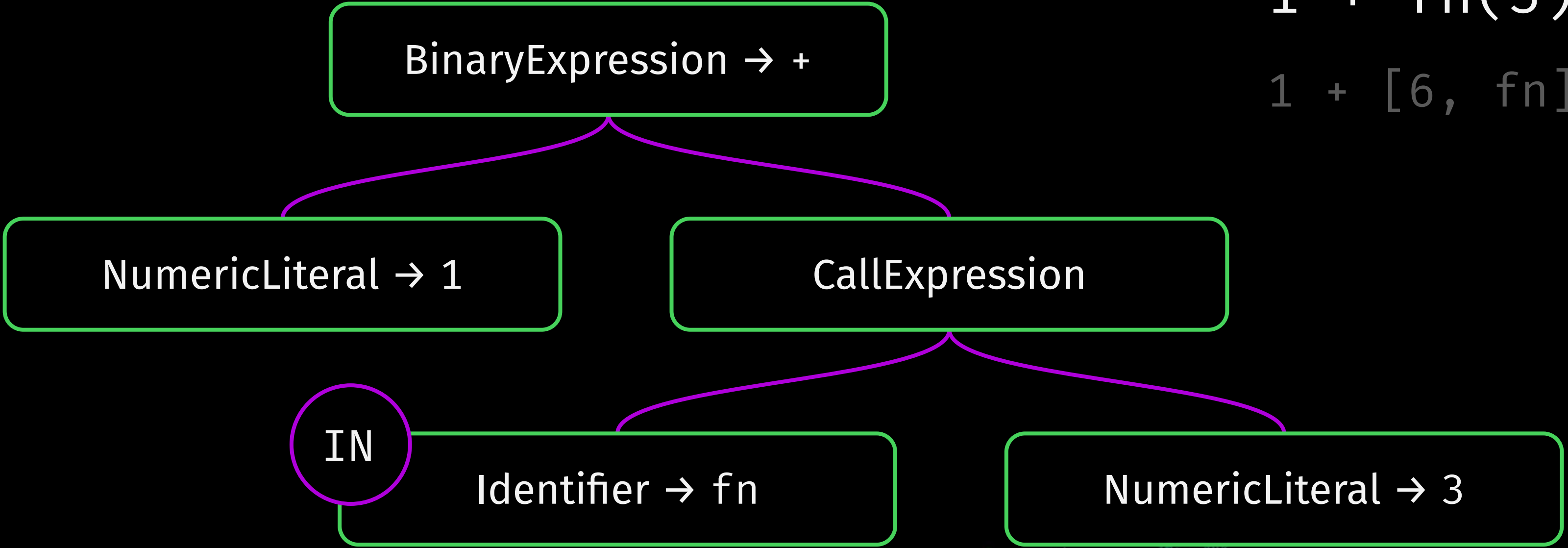


```
1 + fn(3)

1 + [6, fn]
```

@NicoloRibaudo

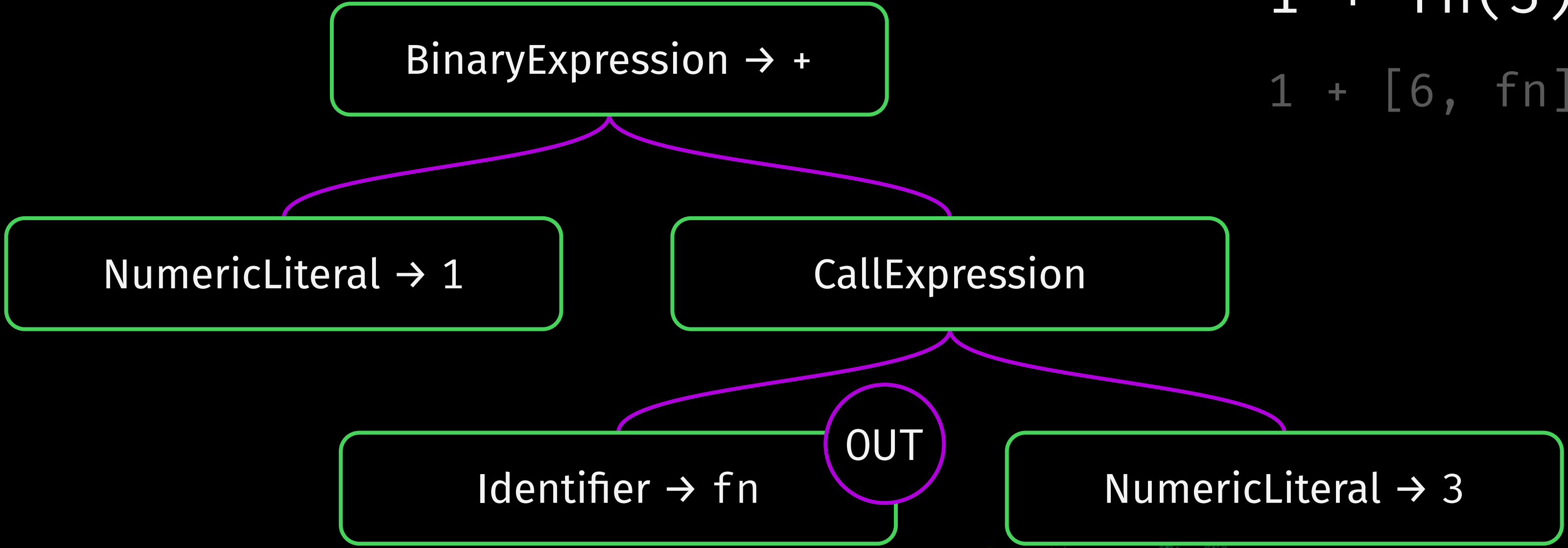
Dynamic Abstract Syntax Tree

```
1 + fn(3)

1 + [6, fn]
```



@NicoloRibaudo

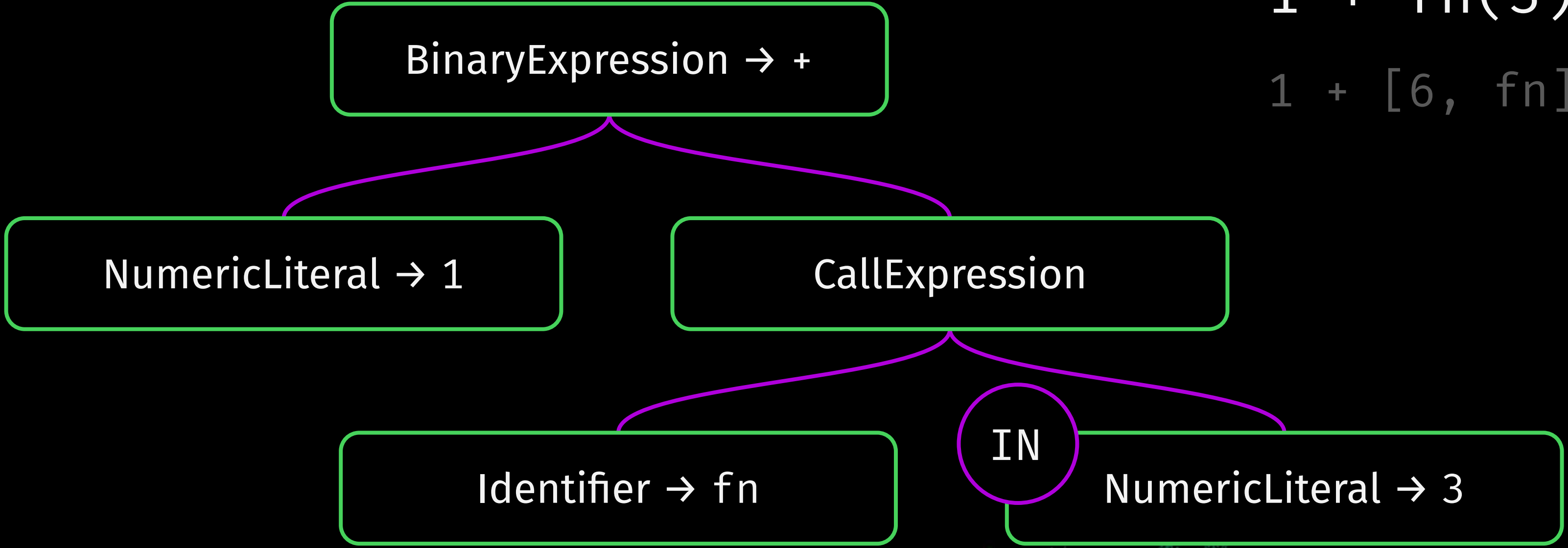
Dynamic Abstract Syntax Tree



```
1 + fn(3)
1 + [6, fn]
```

@NicoloRibaudo

Dynamic Abstract Syntax Tree

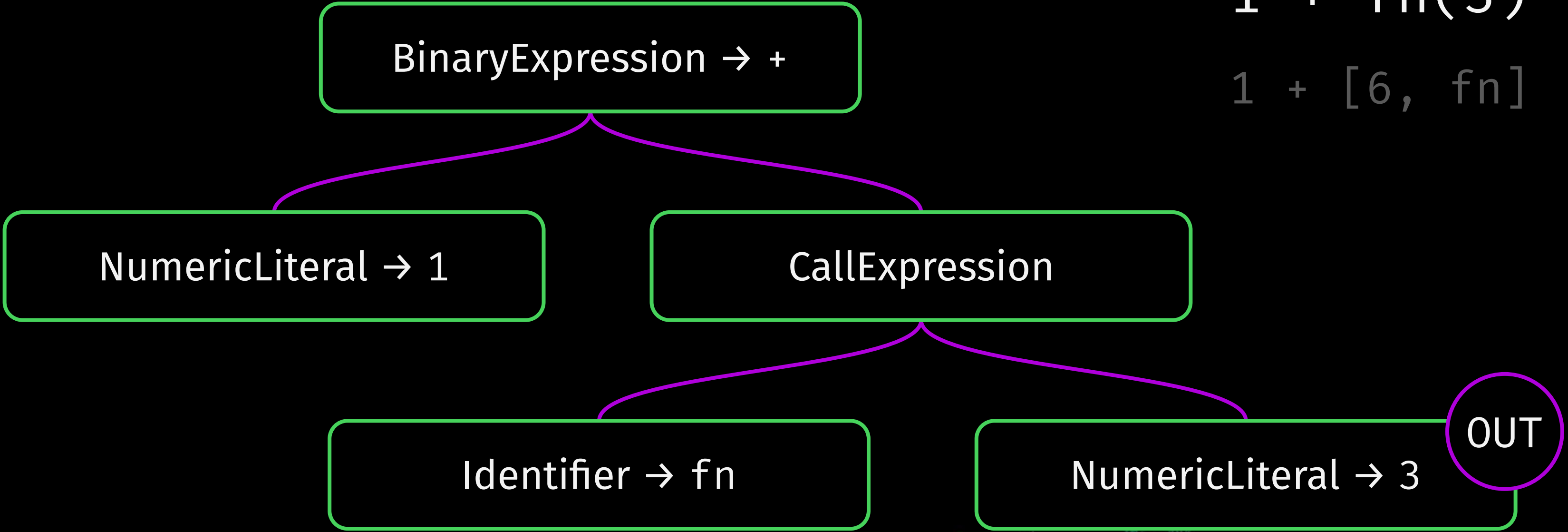


```
1 + fn(3)
1 + [6, fn]
```

@NicoloRibaudo

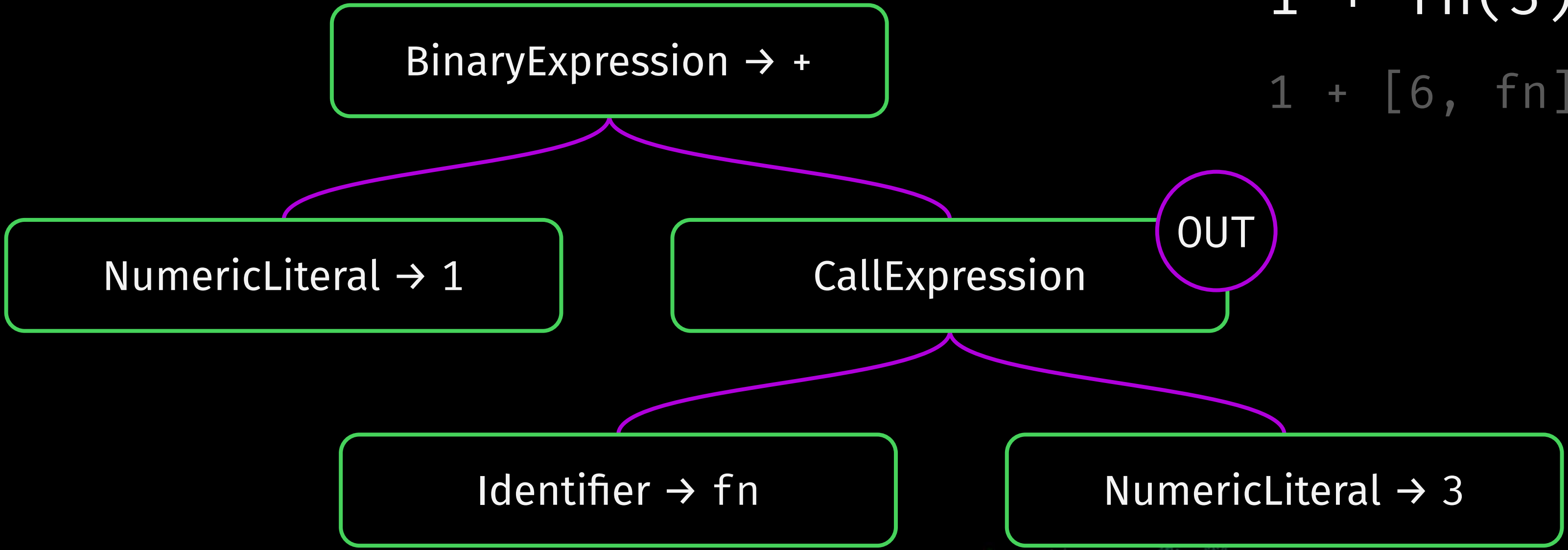
Dynamic Abstract Syntax Tree

```
1 + fn(3)
1 + [6, fn]
```



Dynamic Abstract Syntax Tree

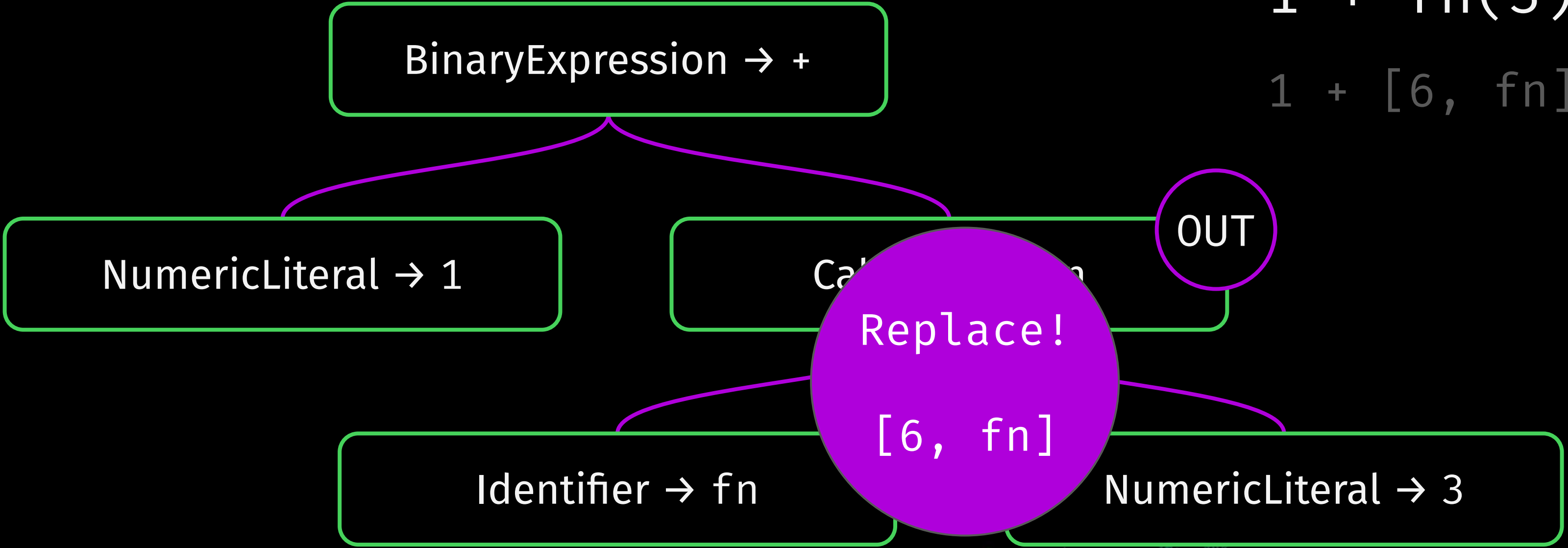
```
1 + fn(3)

1 + [6, fn]
```



@NicoloRibaudo

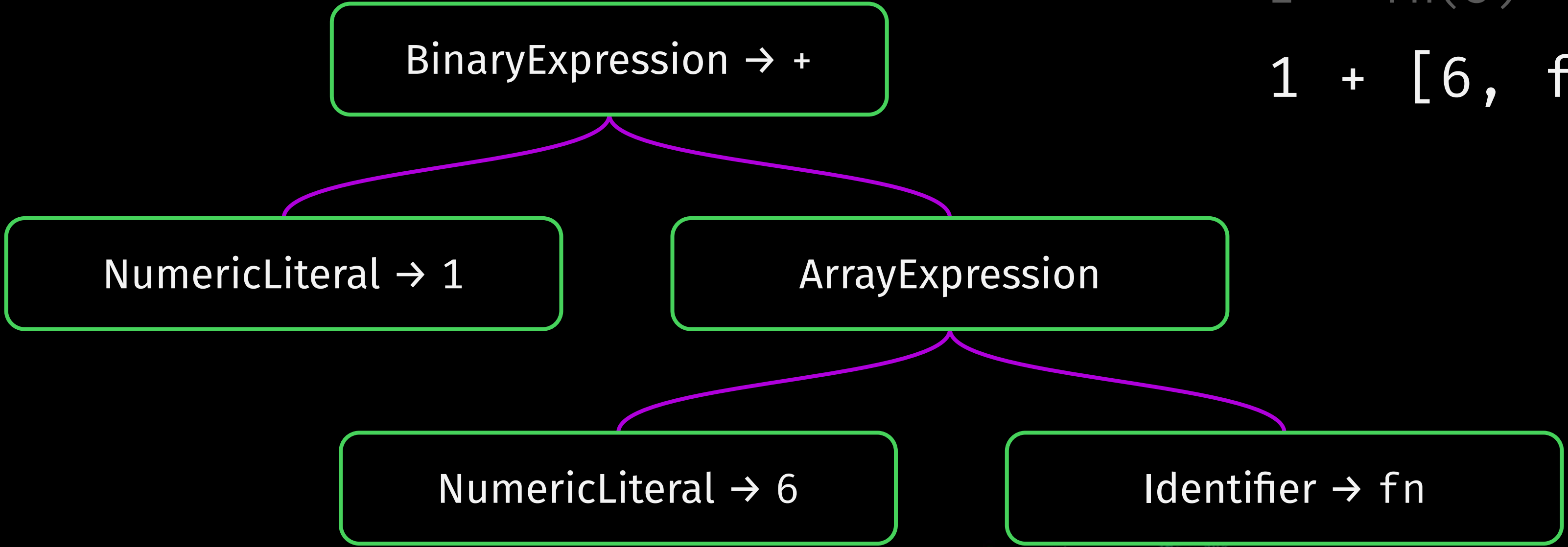
Dynamic Abstract Syntax Tree

```
1 + fn(3)
1 + [6, fn]
```



@NicoloRibaudo

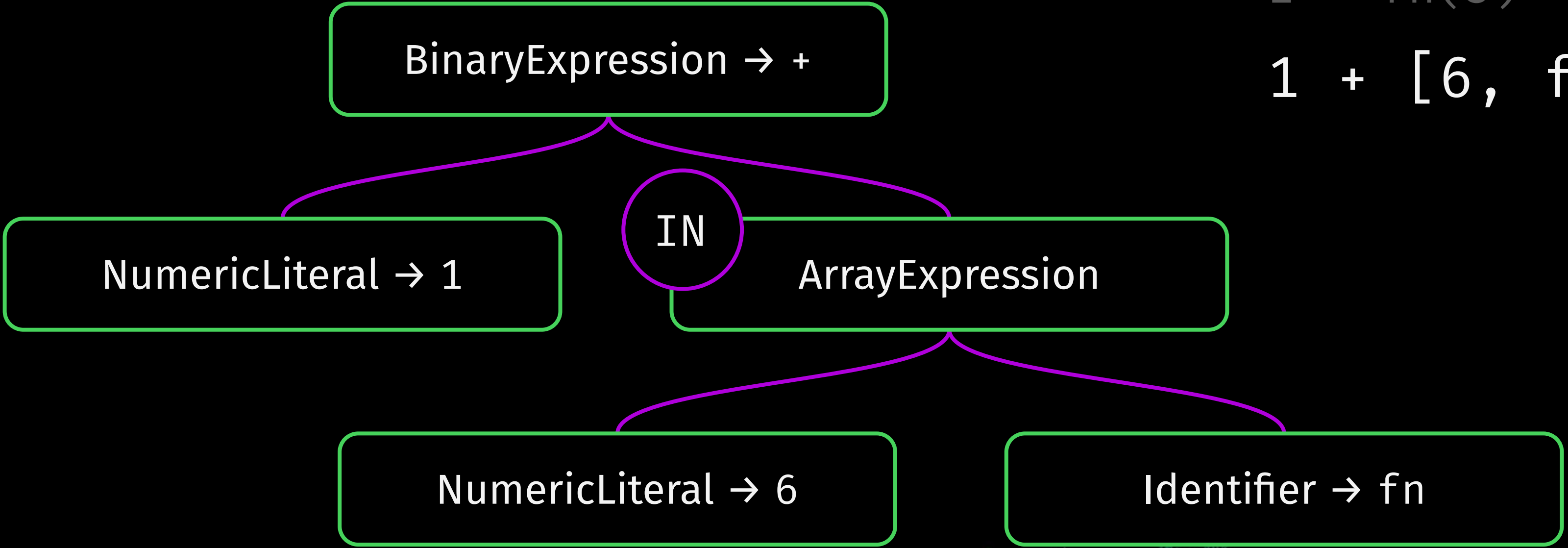
Dynamic Abstract Syntax Tree



```
1 + fn(3)
1 + [6, fn]
```

Dynamic Abstract Syntax Tree

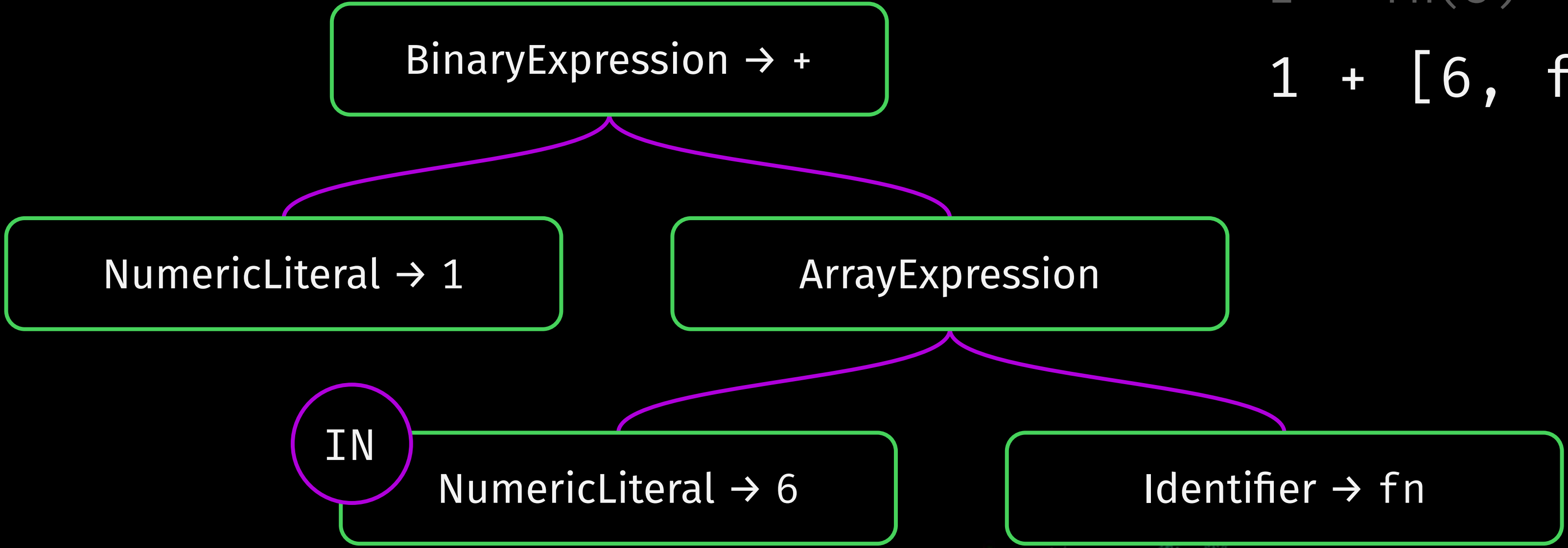
```
1 + fn(3)
1 + [6, fn]
```



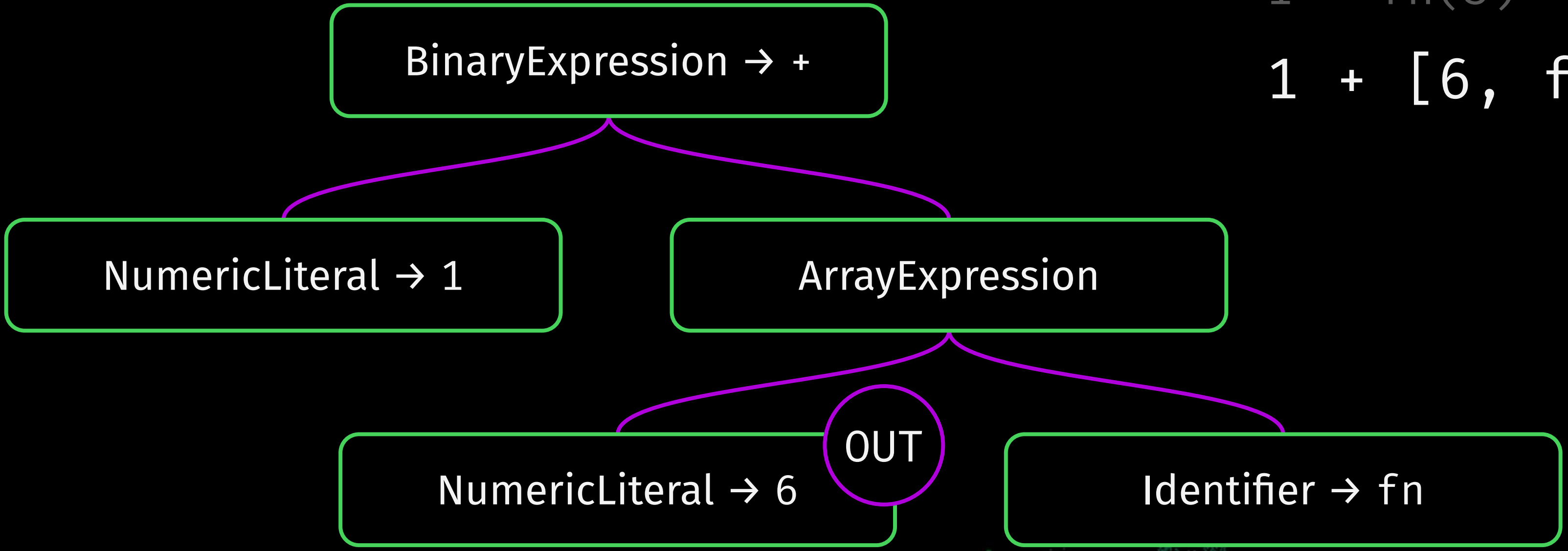
@NicoloRibaudo

Dynamic Abstract Syntax Tree

```
1 + fn(3)
1 + [6, fn]
```



Dynamic Abstract Syntax Tree

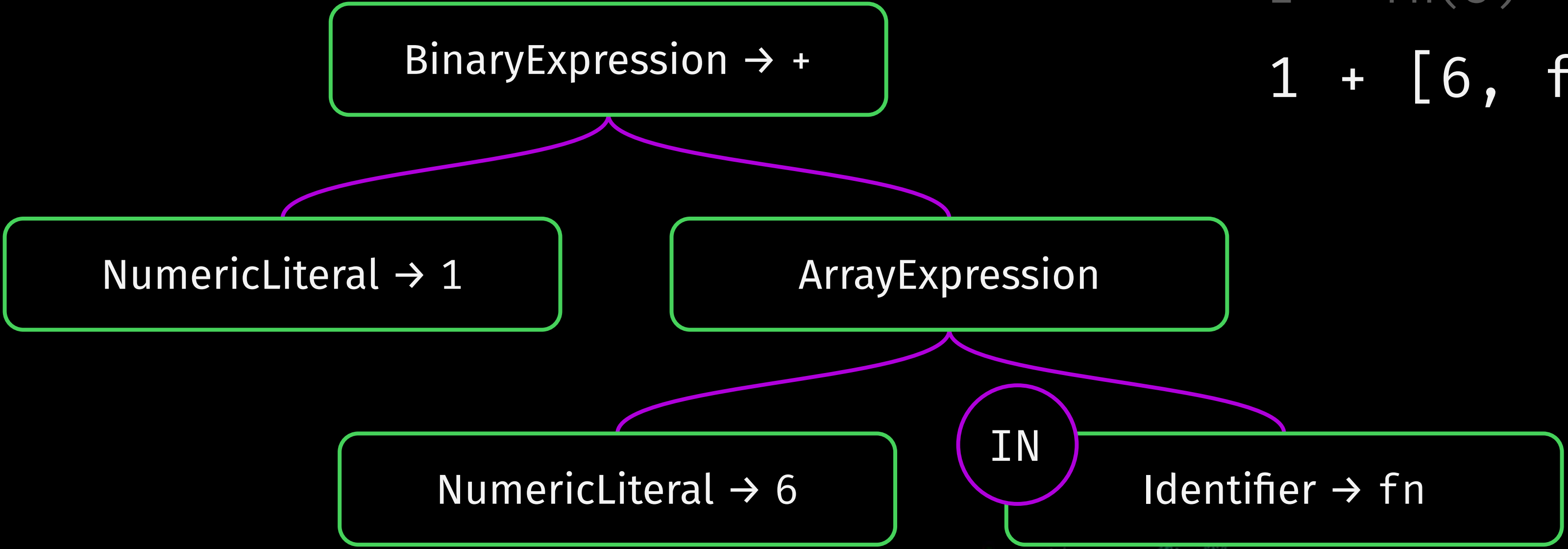


```
1 + fn(3)
1 + [6, fn]
```

@NicoloRibaudo

Dynamic Abstract Syntax Tree

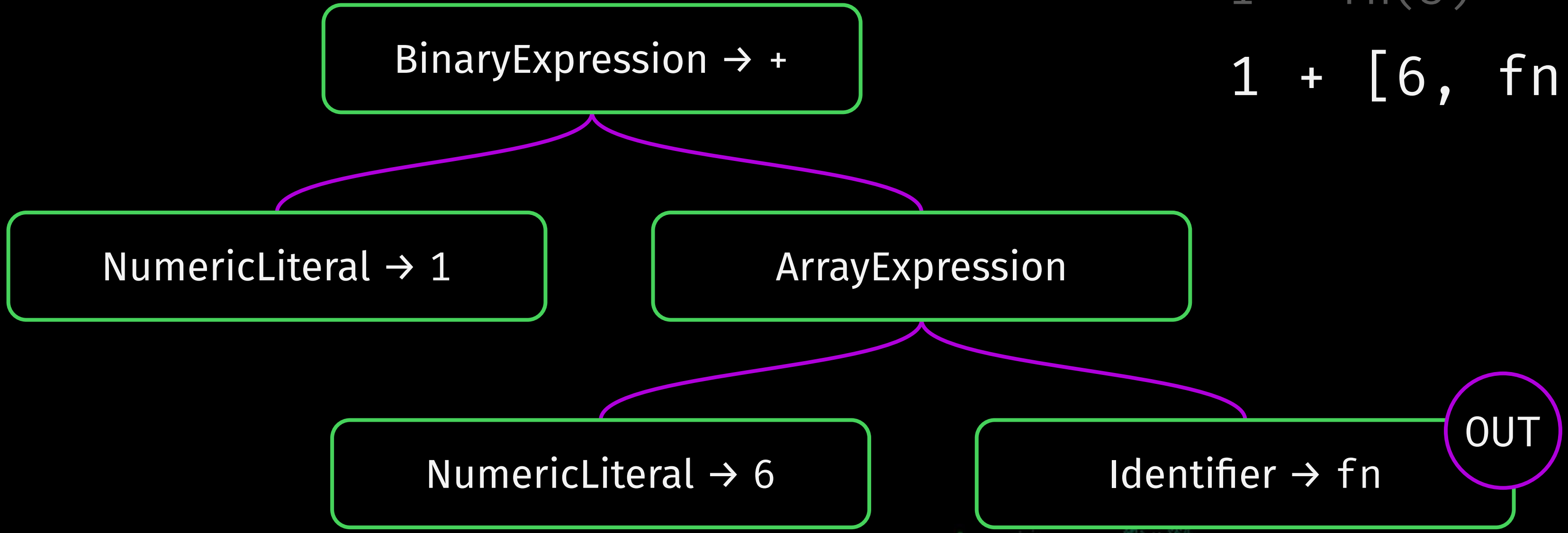
```
1 + fn(3)
1 + [6, fn]
```



@NicoloRibaudo

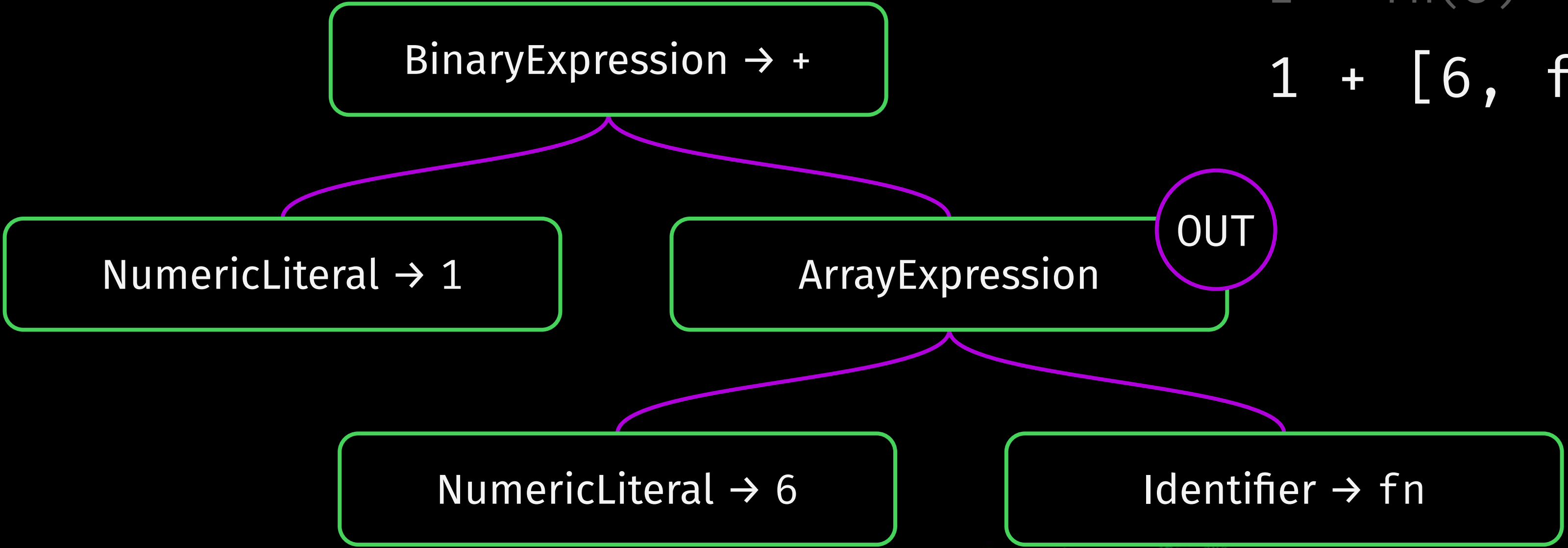
Dynamic Abstract Syntax Tree

```
1 + fn(3)
1 + [6, fn]
```



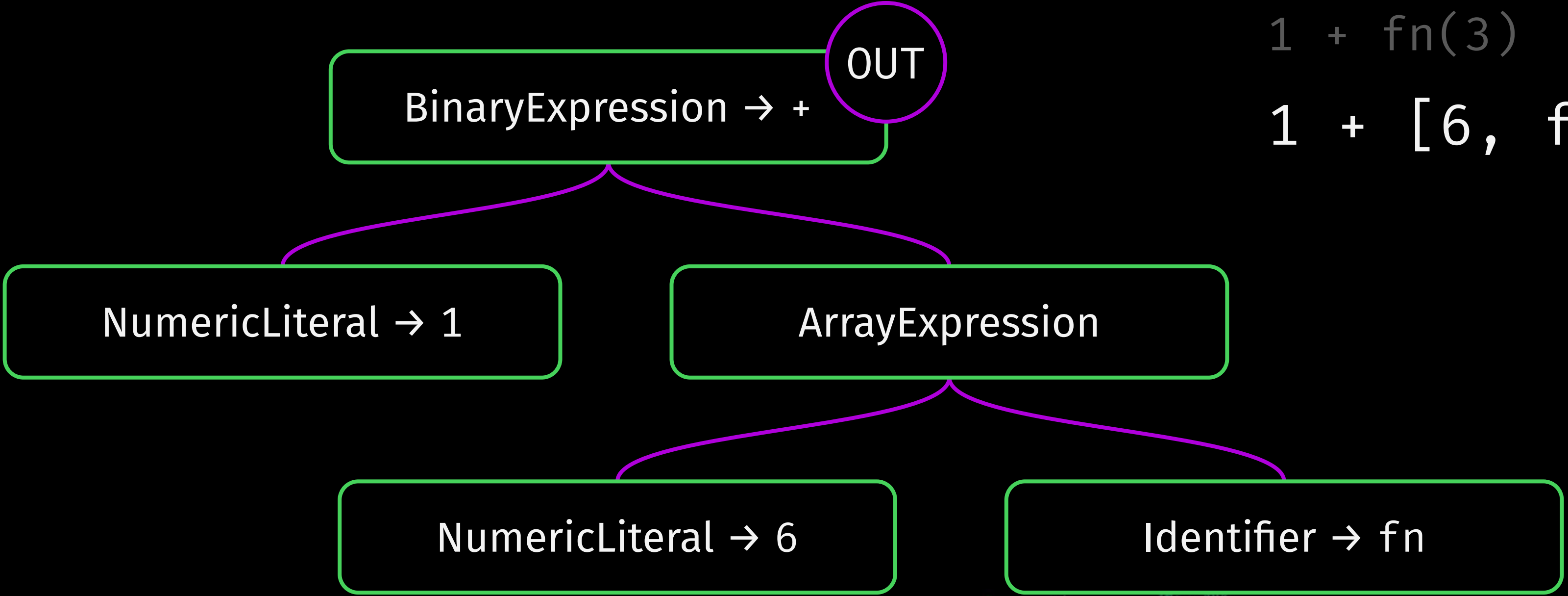
Dynamic Abstract Syntax Tree

```
1 + fn(3)
1 + [6, fn]
```



@NicoloRibaudo

Dynamic Abstract Syntax Tree



```
1 + fn(3)
1 + [6, fn]
```

Utilities

NODES (AST)

Search

Introspection

Evaluation

Insertion

Removal

Replacement

第十四届 D2前端技术论坛

Utilities

NODES (AST)

Search

Introspection

Evaluation

Insertion

Removal

Replacement

BINDINGS (SCOPE)

Validation

Tracking

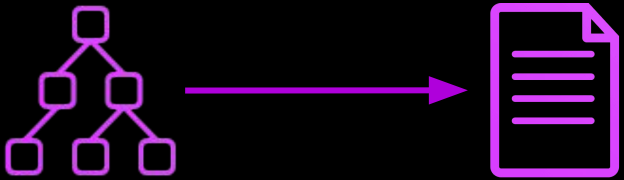
Creation

Renaming

@NicoloRibaudo

A look inside Babel:

`@babel/generator`



Transform AST to source code

Insert parentheses, comments and indentation where needed

Transform AST to source code

Insert parentheses, comments and indentation where needed

⚠️ IT'S NOT A PRETTY PRINTER ⚠️

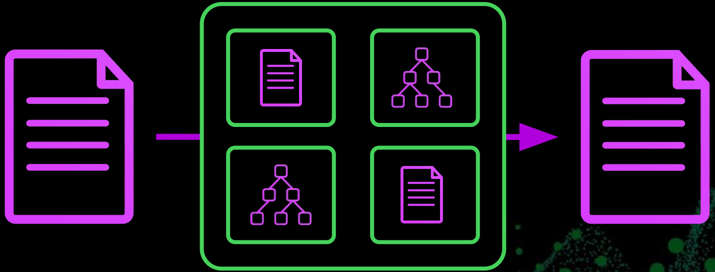
Transform AST to source code

Insert parentheses, comments and indentation where needed

⚠️ IT'S NOT A PRETTY PRINTER ⚠️

A look inside Babel:

@babel/core



Babel's entrypoint

Used by Babel integrations

`@babel/cli`

`@babel/register`

`babel-loader`

`gulp-plugin-babel`

`babelify`

Parcel

Babel's entrypoint

Used by Babel integrations

`@babel/cli`

`@babel/register`

`babel-loader`

`gulp-plugin-babel`

`babelify`

Parcel

Merges configuration sources

`babel.config.js`

`.babelrc`

`package.json`

programmatic options

Babel's entrypoint

Used by Babel integrations

`@babel/cli`

`@babel/register`

`babel-loader`

`gulp-plugin-babel`

`babelify`

Parcel

Merges configuration sources

`babel.config.js`

`.babelrc`

`package.json`

programmatic options

Runs plugins and presets

Bonus package:

`@babel/types`

第十四届
D2前端技术论坛

Nodes validation

@NicoloRibaudo

Nodes validation

Is this node an expression?

```
t.isExpression(node)
```

创心 第十四届 D2前端技术论坛

Nodes validation

Is this node an expression?

```
t.isExpression(node)
```

Is this node an identifier whose name is "test"?

```
t.isIdentifier(node, { name: "test" })
```

Nodes validation

Is this node an expression?

```
t.isExpression(node)
```

Is this node an identifier whose name is "test"?

```
t.isIdentifier(node, { name: "test" })
```

Is this node a sum whose left operand is the child node?

```
t.isBinaryExpression(node, { operator: "+", left: child })
```

Nodes building

Given a `varId` node, how to increment the value it represents by 2?

Nodes building

Given a `varId` node, how to increment the value it represents by 2?

```
{
  type: "AssignmentExpression",
  operator: "+=",
  right: varId,
  left: {
    type: "NumericLiteral",
    value: 2,
  },
}
```

Nodes building

Given a `varId` node, how to increment the value it represents by 2?

```
t.assignmentExpression(
  "+=",
  varId,
  t.numericLiteral(2)
);
```

```
{
  type: "AssignmentExpression",
  operator: "+=",
  right: varId,
  left: {
    type: "NumericLiteral",
    value: 2,
  },
}
```

@NicoloRibaudo

Nodes building

Given a `varId` node, how to increment the value it represents by 2?

```
t.assignmentExpression(
  "+=",
  varId,
  t.numericLiteral(2)
);
```

```
{
  type: "AssignmentExpression",
  operator: "+=",
  right: varId,
  left: {
    type: "NumericLiteral",
    value: 2,
  },
}
```

@NicoloRibaudo

Nodes building

Given a `varId` node, how to increment the value it represents by 2?

```
t.assignmentExpression(
  "+=",
  varId,
  t.numericLiteral(2)
);
```

```
{
  type: "AssignmentExpression",
  operator: "+=",
  right: varId,
  left: {
    type: "NumericLiteral",
    value: 2,
  },
}
```

@NicoloRibaudo

Nodes building

Given a `varId` node, how to increment the value it represents by 2?

```
t.assignmentExpression(
  "+=",
  varId,
  t.numericLiteral(2)
);
```

```
{
  type: "AssignmentExpression",
  operator: "+=",
  right: varId,
  left: {
    type: "NumericLiteral",
    value: 2,
  },
}
```

Nodes building

Given a `varId` node, how to increment the value it represents by 2?

```
t.assignmentExpression(
  "+=",
  varId,
  t.numericLiteral(2)
);
```

```
{
  type: "AssignmentExpression",
  operator: "+=",
  right: varId,
  left: {
    type: "NumericLiteral",
    value: 2,
  },
}
```

Nodes building

Given a `varId` node, how to increment the value it represents by 2?

```
t.assignmentExpression(
  "+=",
  varId,
  t.numericLiteral(2)
);
```

```
{
  type: "AssignmentExpression",
  operator: "+=",
  right: varId,
  left: {
    type: "NumericLiteral",
    value: 2,
  },
}
```

Bonus package:

`@babel/template`

High level AST building

Given a `varId` node referencing an array, how to increment each of its elements by 2 and then take only the values greater than 10?

High level AST building

Given a `varId` node referencing an array, how to increment each of its elements by 2 and then take only the values greater than 10?

```
t.assignmentExpression("=", varId, t.callExpression(
  t.memberExpression(t.callExpression(
    t.memberExpression(varId, t.identifier("map")),
    [t.arrowFunctionExpression([t.identifier("val")],
                           t.binaryExpression("+", t.identifier("val"), t.numericLiteral(2))
    )]
  ), t.identifier("filter")),
  [t.arrowFunctionExpression([t.identifier("val")],
                           t.binaryExpression(">", t.identifier("val"), t.numericLiteral(10))
   )]
))
```

High level AST building

Given a `varId` node referencing an array, how to increment each of its elements by 2 and then take only the values greater than 10?

```
template.expression.ast`
   ${varId} = ${varId}
                 .map(val => val + 2)
                 .filter(val => val > 10)
`
```

第十四届
D2前端技术论坛

High level AST building

Different parsing goals

```
template.expression

template.statement

template.statements

template.program
```

@NicoloRibaudo

High level AST building

Different parsing goals

```
template.expression

template.statement

template.statements

template.program
```

Immediate usage...

```
ast = template.*.ast`${val} * 2`
```

... or deferred usage

```
build = template.*(`%%val%% * 2`)
// ...
ast = build({ val })
```

Plugins

Everything is a plugin

Everything is a plugin

ECMAScript features	`@babel/plugin-transform-classes`

Everything is a plugin

ECMAScript features	`@babel/plugin-transform-classes`
ECMAScript proposals	`@babel/plugin-proposal-private-methods`

Everything is a plugin

ECMAScript features	`@babel/plugin-transform-classes`
ECMAScript proposals	`@babel/plugin-proposal-private-methods`
ECMAScript extensions	`@babel/plugin-transform-typescript` `@babel/plugin-transform-react-jsx`

Everything is a plugin

ECMAScript features	`@babel/plugin-transform-classes`
ECMAScript proposals	`@babel/plugin-proposal-private-methods`
ECMAScript extensions	`@babel/plugin-transform-typescript` `@babel/plugin-transform-react-jsx`
Optimization	`@babel/plugin-transform-runtime`

Everything is a plugin

babel-plugin-module-resolver

babel-plugin-macros

babel-plugin-transform-define

babel-plugin-emotion

babel-plugin-inferno

babel-plugin-add-module-exports

babel-plugin-istanbul

babel-plugin-react-css-modules

babel-plugin-react-intl-auto

babel-plugin-transform-async-to-promises

How to create a plugin

1. Create a function

```
function myPlugin(babel, options) {
  return {
    name: "my-plugin",
    visitor: {
      CallExpression(path) { /* ... */ }
    },
    manipulateOptions(babelOptions) {},
    inherits: require("another-plugin"),
  };
}
```

@NicoloRibaudo

1. Create a function

```
function myPlugin(babel, options) {
  return {
    name: "my-plugin",
    visitor: {
      CallExpression(path) { /* ... */ }
    },
    manipulateOptions(babelOptions) {},
    inherits: require("another-plugin"),
  };
}
```

The first parameter exposes all the public API and utilities

```
// @babel/types
const t = babel.types;
```

@NicoloRibaudo

1. Create a function

```
function myPlugin(babel, options) {
  return {
    name: "my-plugin",
    visitor: {
      CallExpression(path) { /* ... */ }
    },
    manipulateOptions(babelOptions) {},
    inherits: require("another-plugin"),
  };
}
```

The second parameter contains the options for this plugin defined in the user's config

@NicoloRibaudo

2. Choose a name

Required

```
function myPlugin(babel, options) {
  return {
    name: "my-plugin",
    visitor: {
      CallExpression(path) { /* ... */ }
    },
    manipulateOptions(babelOptions) {},
    inherits: require("another-plugin"),
  };
}
```

Should match the plugin package name

`babel-plugin-my-plugin`

@NicoloRibaudo

3. Define traversal visitor

Optional

```
function myPlugin(babel, options) {
  return {
    name: "my-plugin",
    visitor: {
      CallExpression(path) { /* ... */ }
    },
    manipulateOptions(babelOptions) {},
    inherits: require("another-plugin"),
  };
}
```

4. Modify Babel options

Optional

```
function myPlugin(babel, options) {
  return {
    name: "my-plugin",
    visitor: {
      CallExpression(path) { /* ... */ }
    },
    manipulateOptions(babelOptions) {},
    inherits: require("another-plugin"),
  };
}
```

It also handles options for `@babel/parser` and `@babel/generator`

```
opts.parserOpts
opts.generatorOprs
```

5. Extend another plugin

Optional

```
function myPlugin(babel, options) {
  return {
    name: "my-plugin",
    visitor: {
      CallExpression(path) { /* ... */ }
    },
    manipulateOptions(babelOptions) {},
    inherits: require("another-plugin"),
  };
}
```

How to create a plugin

```
function myPlugin(babel, options) {
  return {
    name: "my-plugin",
    visitor: {
      CallExpression(path) { /* ... */ }
    },
    manipulateOptions(babelOptions) {},
    inherits: require("another-plugin"),
  };
}
```

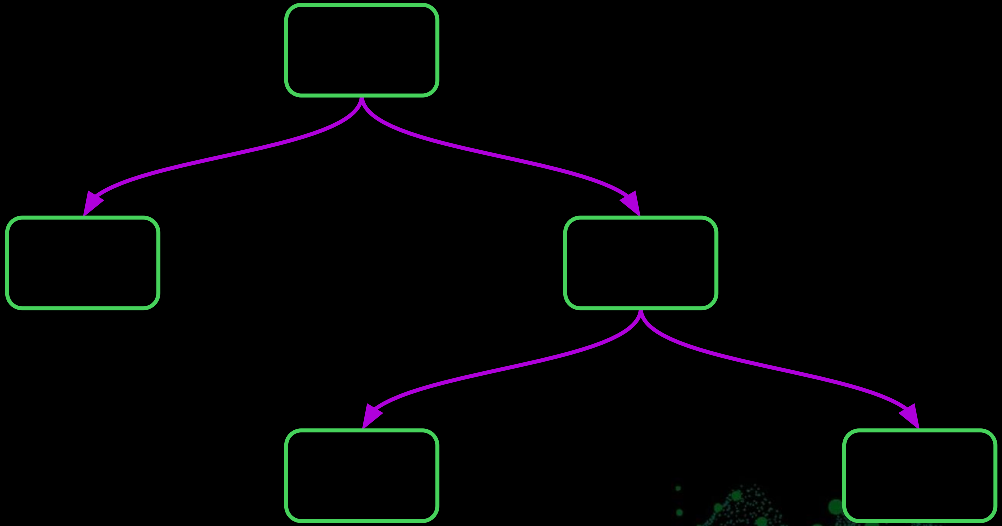
@NicoloRibaudo

A node with superpowers:

NodePath

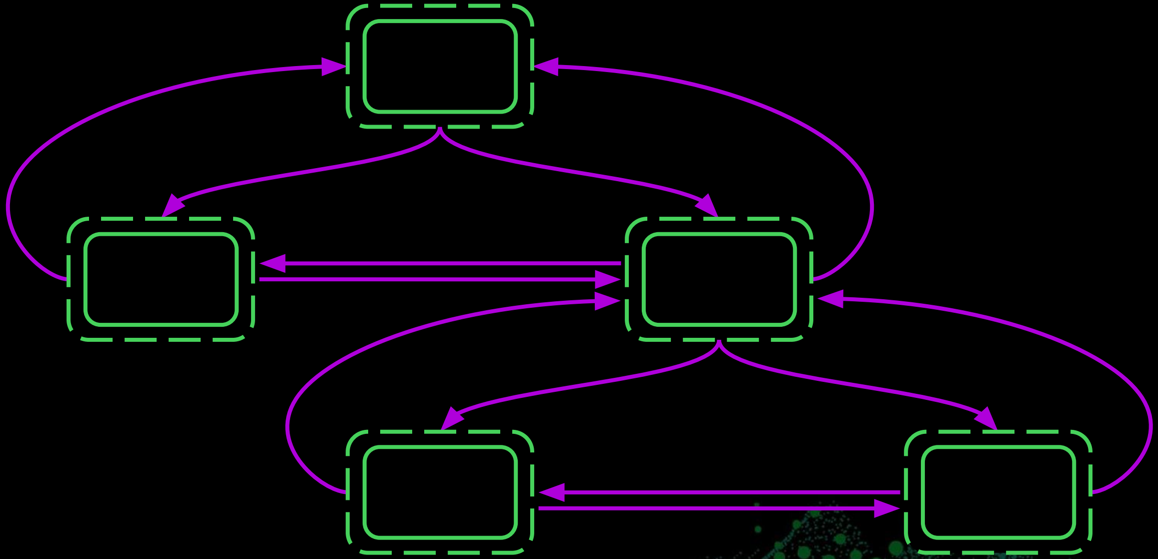
NodePath

Transformations need *context* and *ergonomics* for AST manipulation



NodePath

Transformations need *context* and *ergonomics* for AST manipulation



@NicoloRibaudo

NodePath

NodePath

`path.node` Get the original, unwrapped, node

NodePath

`path.node`	Get the original, unwrapped, node
`path.parentPath`	Get the path of parent node ...
`path.get("body.0.id")`	.. or of a child node

@NicoloRibaudo

NodePath

`path.node`	Get the original, unwrapped, node
`path.parentPath`	Get the path of parent node ...
`path.get("body.0.id")`	.. or of a child node
`path.scope`	Get the scope of the current path

NodePath

`path.node`	Get the original, unwrapped, node
`path.parentPath`	Get the path of parent node ...
`path.get("body.0.id")`	.. or of a child node
`path.scope`	Get the scope of the current path
`path.replaceWith(node)`	Replace the current node with another one ...
`path.insertBefore(...nodes)`	... or just insert some new nodes before ...
`path.insertAfter(...nodes)`	... or after

NodePath

`path.node`	Get the original, unwrapped, node
`path.parentPath`	Get the path of parent node ...
`path.get("body.0.id")`	.. or of a child node
`path.scope`	Get the scope of the current path
`path.replaceWith(node)`	Replace the current node with another one ...
`path.insertBefore(...nodes)`	... or just insert some new nodes before ...
`path.insertAfter(...nodes)`	... or after
`path.toString()`	Call `@babel/generator`, useful when debugging

Case study

throw expressions

```
name || throw new Error()
```

Cas udy

throw sions

`name l rror()`

@NicoloRibaudo

`throw` expressions

Allow using `throw` wherever an expression can be used:

throw expressions

Allow using `throw` wherever an expression can be used:

```
var num = typeof input === "string" ? parseInt(input)
        : typeof input === "number" ? input
        : throw new Error("input must be a number or a string");
```

throw expressions

Allow using throw wherever an expression can be used:

```
var num = typeof input === "string" ? parseInt(input)
       : typeof input === "number" ? input
       : throw new Error("input must be a number or a string");

function double(x = throw new Error("x is required")) {
   return x * 2;
}
```

@NicoloRibaudo

第十四届
D2前端技术论坛

`throw` expressions

They can be transformed using an *`II=>FE`* *

* *Immediately Invoked Arrow Function Expression*

@NicoloRibaudo

throw expressions

They can be transformed using an *II=>FE* *

```
var x =           throw new Error("Err!")      ;
```

* *Immediately Invoked Arrow Function Expression*

@NicoloRibaudo

throw expressions

They can be transformed using an *II=>FE* *

```
var x =           throw new Error("Err!")       ;

  ↓    ↓    ↓    ↓    ↓    ↓    ↓    ↓

var x = (() => { throw new Error("Err!"); })();
```

* *Immediately Invoked Arrow Function Expression*

@NicoloRibaudo

babel-plugin-throw-expressions

babel-plugin-throw-expressions

```
export default function plugin() {

      /* ... */

      /* ... */



      /* ... */

}
```

babel-plugin-throw-expressions

```
export default function plugin() {
    return {
        /* ... */

        /* ... */




        /* ... */
    };
}
```

@NicoloRibaudo

babel-plugin-throw-expressions

```
export default function plugin() {
    return {
        name: "throw-expressions",

        /* ... */




        /* ... */
    };
}
```

@NicoloRibaudo

创心 第十四届 D2前端技术论坛

babel-plugin-throw-expressions

```
export default function plugin() {
    return {
        name: "throw-expressions",

        manipulateOptions(opts) {
            opts.parserOpts.plugins.push("throwExpressions");
        },

        /* ... */
    };
}
```

@NicoloRibaudo

babel-plugin-throw-expressions

```
export default function plugin() {
    return {
        name: "throw-expressions",

        manipulateOptions(opts) {
            opts.parserOpts.plugins.push("throwExpressions");
        },

        visitor: { /* ... */ }
    };
}
```

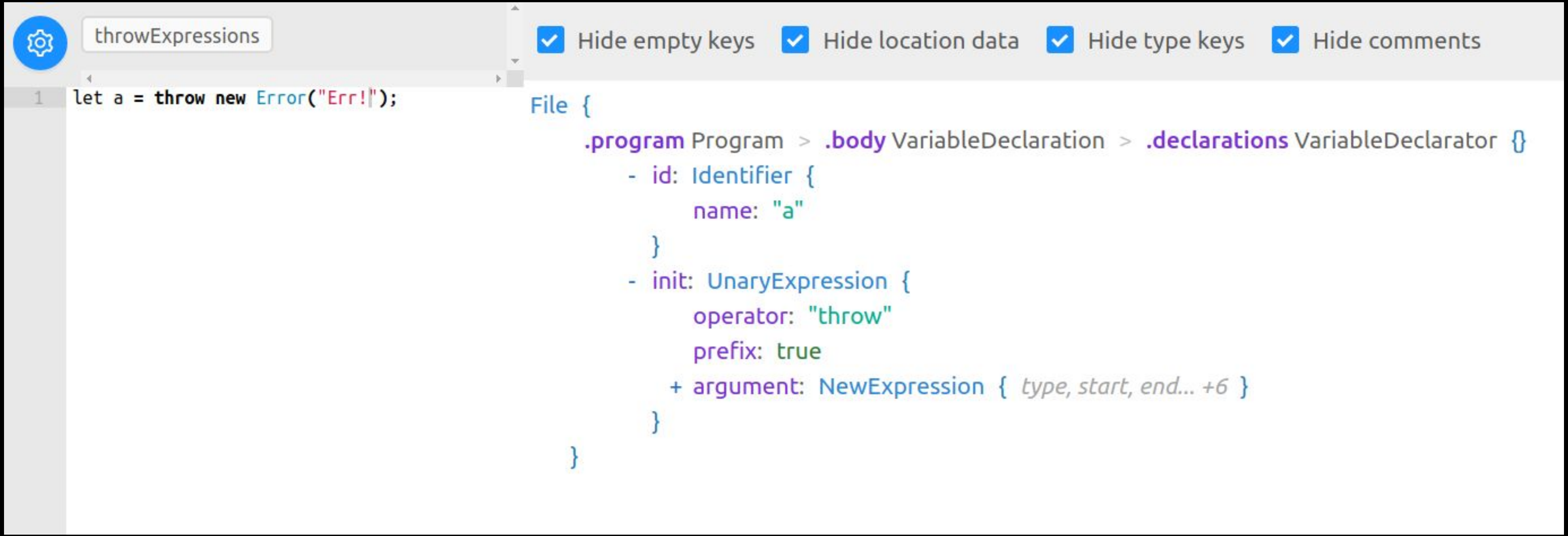
@NicoloRibaudo

babel-plugin-throw-expressions

```
visitor: {

      /* ... */


      /* ... */



      /* ... */

},
```

@NicoloRibaudo

babel-plugin-throw-expressions



```
let a = throw new Error("Err!");
```

```
File {
    .program Program > .body VariableDeclaration > .declarations VariableDeclarator {}
        - id: Identifier {
              name: "a"
          }
        - init: UnaryExpression {
              operator: "throw"
              prefix: true
            + argument: NewExpression { type, start, end... +6 }
          }
    }
```

https://lihautan.com/babel-ast-explorer https://astexplorer.net/

@NicoloRibaudo

babel-plugin-throw-expressions

```
visitor: {
    UnaryExpression(path) {
        /* ... */



        /* ... */





        /* ... */
    },
},
```

@NicoloRibaudo

babel-plugin-throw-expressions

```
visitor: {
    UnaryExpression(path) {
        const { node } = path;
        if (node.operator !== "throw") return;

        /* ... */




        /* ... */
    },
},
```

babel-plugin-throw-expressions

```
visitor: {
    UnaryExpression(path) {
        const { node } = path;
        if (node.operator !== "throw") return;

        const iife = template.expression.ast`
            (() => { throw ${node.argument}; })()
        `;

        /* ... */
    },
},
```

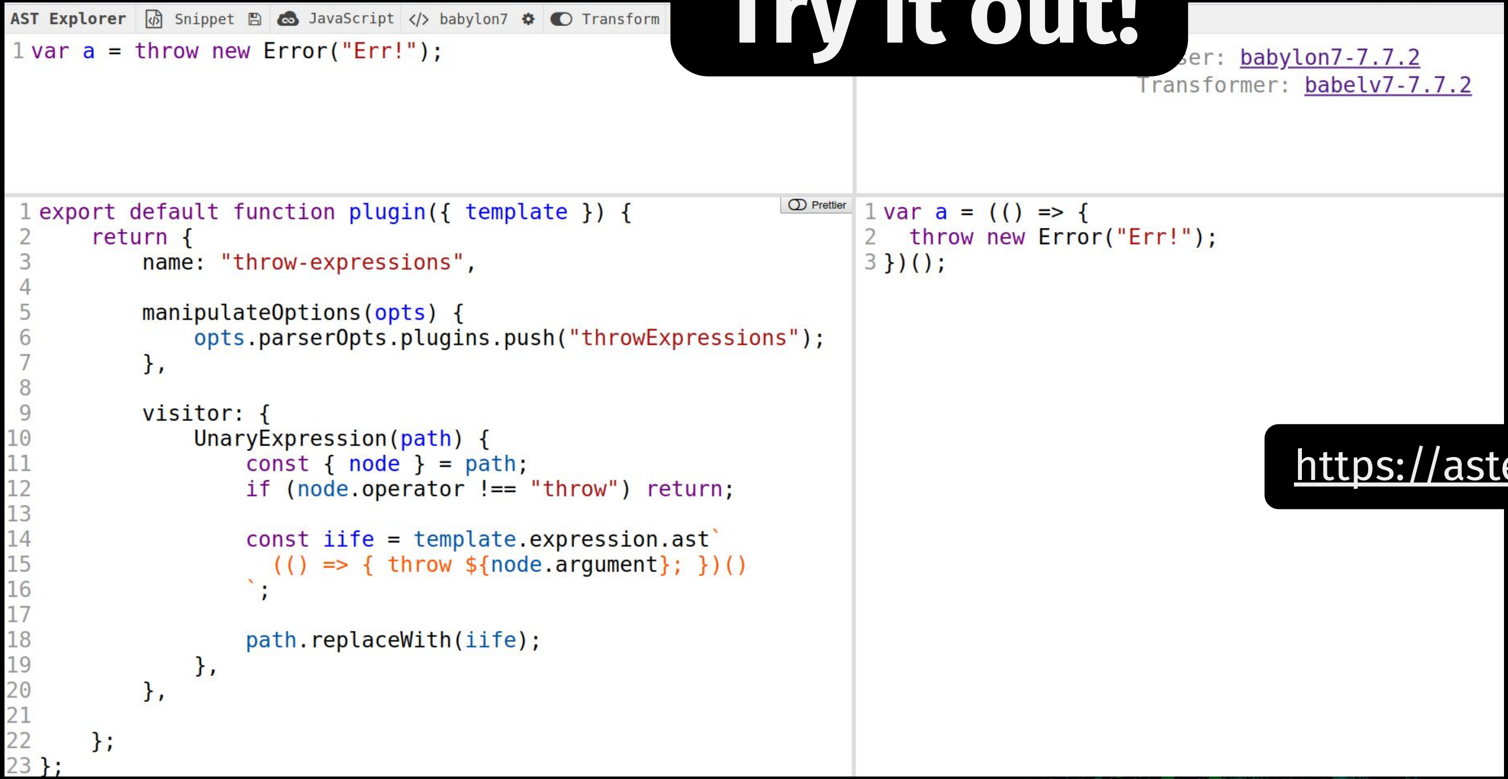
@NicoloRibaudo

babel-plugin-throw-expressions

```
visitor: {
    UnaryExpression(path) {
        const { node } = path;
        if (node.operator !== "throw") return;

        const iife = template.expression.ast`
            (() => { throw ${node.argument}; })()
        `;

        path.replaceWith(iife);
    },
},
```

@NicoloRibaudo

Try it out!



```
var a = throw new Error("Err!");
```

Parser: babylon7-7.7.2
Transformer: babelv7-7.7.2

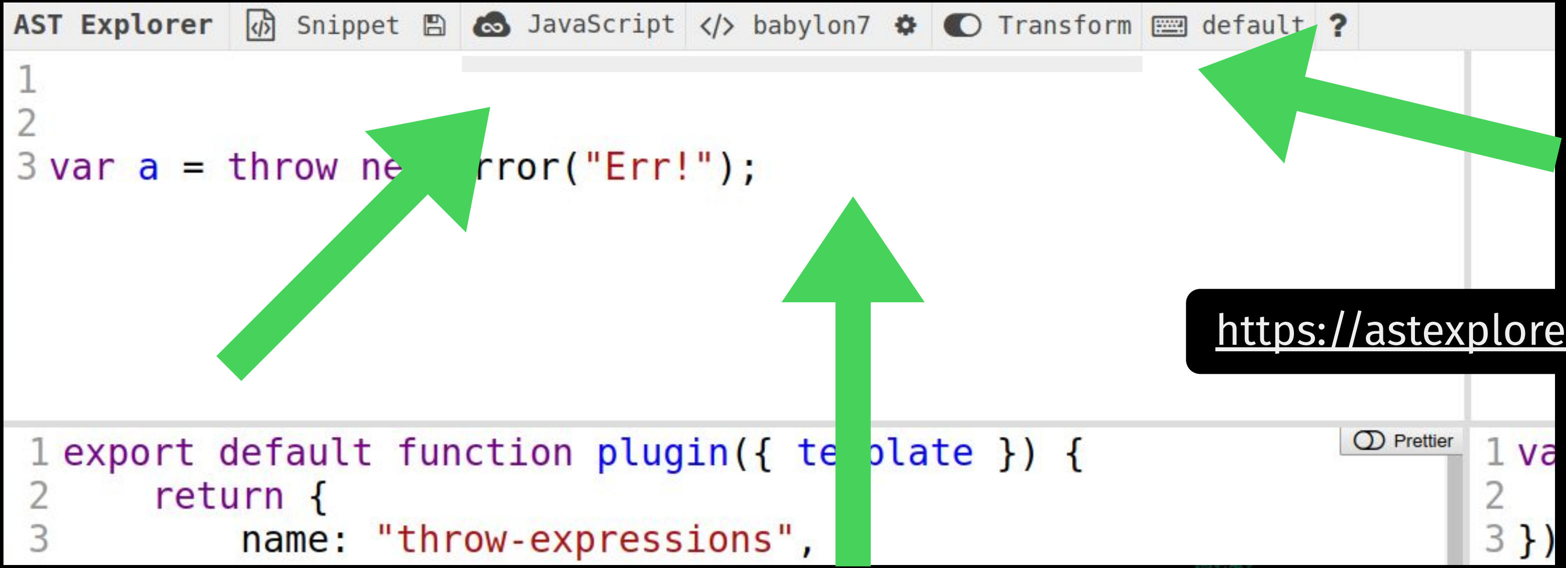
```
export default function plugin({ template }) {
    return {
        name: "throw-expressions",

        manipulateOptions(opts) {
            opts.parserOpts.plugins.push("throwExpressions");
        },

        visitor: {
            UnaryExpression(path) {
                const { node } = path;
                if (node.operator !== "throw") return;

                const iife = template.expression.ast`
                  (() => { throw ${node.argument}; })()
                `;

                path.replaceWith(iife);
            },
        },

    };
};
```

```
var a = (() => {
  throw new Error("Err!");
})();
```

https://astexplorer.net/

@NicoloRibaudo

Try it out!



```
var a = throw ne    rror("Err!");
```

```
export default function plugin({ te  plate }) {
    return {
        name: "throw-expressions",
```

```
va
})
```

https://astexplorer.net/

结束

结束？

One more thing ...

One more thing ...

Babel is a community based project:
it is not developed by a company.

One more thing ...

Babel is a community based project:
it is not developed by a company.

Babel's future and sustainability is guaranteed
thanks to the donations made by our users.

One more thing ...

Babel is a community based project:
it is not developed by a company.

Babel's future and sustainability is guaranteed thanks to the donations made by our users.

If your company uses Babel and you could be interested in sponsoring us, please get in touch!

NICOLÒ RIBAUDO

Babel team

@NicoloRibaudo

@nicolo-ribaudo

@nicolo-ribaudo

nicolo.ribaudo@gmail.com

NICOLÒ RIBAUDO

Babel team

@NicoloRibaudo

@nicolo-ribaudo

@nicolo-ribaudo

nicolo.ribaudo@gmail.com

@nicolo-ribaudo

Feedback