

How works Babel?

Sven Sauleau

2018



@svensauleau

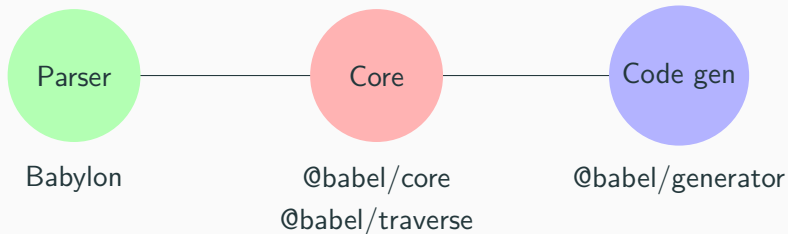
BABEL



$$Source = \left\{ \begin{array}{c} \text{JSX} \\ \text{ES2015} \\ \vdots \end{array} \right\}$$

$$Babel = Source \rightarrow Source'$$

Babel toolchain



$Parser = Source_1 \rightarrow AST_1$

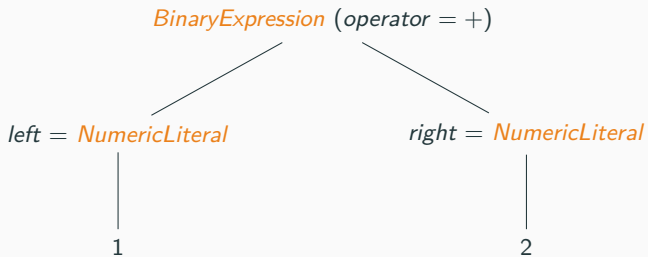
$Core = AST_1 \rightarrow AST_2$

$Codegen = AST_2 \rightarrow Source_2$

Abstract Syntax Tree.

It's a representation for computers of your program.

1 + 2



@babel/types¹ aka *t*.

AST builder methods.

```
s = t.stringLiteral("test")
```

```
b = t.booleanLiteral(false)
```

```
w = t.identifier("window")
```

¹<https://github.com/babel/babel/tree/master/packages/babel-types>

@babel/template²

AST from a string template.

```
1 const buildLog = template(`  
2   console.log(TEXT)  
3 `);  
4  
5 const ast = buildLog({  
6   TEST: t.stringLiteral("Hi")  
7 });
```

²<https://github.com/babel/babel/tree/master/packages/babel-template>

@babel/traverse³

Traversal/manipulation of the AST.

```
1 traverse(ast, {  
2   // visitors  
3 });
```

³<https://github.com/babel/babel/tree/master/packages/babel-traverse>

Visitor pattern

“[...] the visitor design pattern is a way of separating an algorithm from an object structure on which it operates.”

— https://en.wikipedia.org/wiki/Visitor_pattern

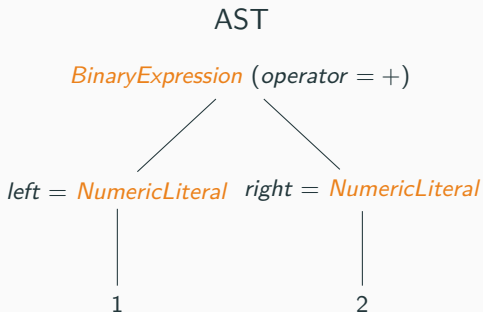
Allows to visit AST nodes.

Example

BinaryExpression Visitor

Visitors

```
1 BinaryExpression({node}) {  
2   node.operator = '-';  
3 }
```

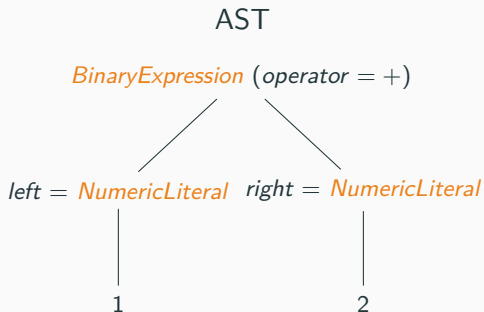


$1 + 2 \rightarrow 1 - 2$

NumericLiteral Visitor

Visitors

```
1 NumericLiteral({node}) {  
2   if (node.value === 1) {  
3     node.value = 2;  
4   }  
5 },
```



$1 + 2 \rightarrow 2 + 2$

Path manipulation: remove

```
1 BooleanLiteral(path) {  
2  
3     path.remove();  
4  
5 }
```

In

var bar;

true;

Out

var bar;

Path manipulation: replaceWith

```
1 BooleanLiteral(path) {  
2  
3     path.replaceWith(  
4         /* replacement */ t.identifier("bar")  
5     );  
6  
7 }
```

In

true;

Out

bar;

Path manipulation: replaceWithString

```
1 BooleanLiteral(path) {  
2  
3     path.replaceWithString(  
4         "(function () { var a = 1 })()"  
5     );  
6  
7 }
```

In

true;

Out

```
1 (function () {  
2     var a = 1;  
3 })();
```


Path/scope manipulation: rename

```
1 FunctionDeclaration(path) {  
2  
3     path.scope.rename(  
4         /* oldName */ "a",  
5         /* newName? */ "renamed_a"  
6     );  
7 }
```

In

```
1 function t() {  
2     var a;  
3     a;  
4     a;  
5 }  
6  
7 a;
```

Out

```
1 function t() {  
2     var renamed_a;  
3     renamed_a;  
4     renamed_a;  
5 }  
6  
7 a;
```

Path manipulation: insertBefore

4

```
1 BooleanLiteral(path) {  
2  
3     const node = t.stringLiteral("foo");  
4     path.insertBefore(node);  
5  
6 },
```

In

```
1 true;
```

Out

```
1 "foo";  
2 true;
```

⁴same with after

Path/scope manipulation: hoist

```
1 BooleanLiteral(path) {  
2     path.hoist();  
3 },
```

In

```
1  
2  
3 function t() {  
4     var a = true;  
5 }
```

Out

```
1 var _ref = true;  
2  
3 function t() {  
4     var a = _ref;  
5 }
```

Path manipulation: matchesPattern

```
1 MemberExpression ( path ) {  
2  
3     if ( path.matchesPattern( "foo.bar" ) ) {  
4         path.remove();  
5     }  
6  
7 }
```

In

```
1 foo ;  
2 foo . bar ;  
3 foo [ 'bar' ] ;
```

Out

```
1 foo ;
```

Path manipulation: evaluate

```
1 BinaryExpression(path) {  
2  
3     const {value} = path.evaluate();  
4     path.replaceWith(t.numericLiteral(value));  
5  
6 }
```

$$1 + 1 \rightarrow 2$$

traversal: stop

Maximum call stack size exceeded

```
1 BooleanLiteral(path) {  
2     path.replaceWith(t.BooleanLiteral(true));  
3 }
```

OK

```
1 BooleanLiteral(path) {  
2     path.replaceWith(t.BooleanLiteral(true));  
3     path.stop();  
4 }
```

Babel plugin

```
1 module.exports = function({ types: t }) {  
2   return {  
3     visitor: {  
4       // visitors  
5     },  
6   };  
7 };
```

Babel preset

```
1 function myPlugin({ types: t }) {  
2   return {  
3     visitor: {  
4       // visitors  
5     },  
6   };  
7 };  
8  
9 module.exports = function () {  
10   return {  
11     plugins: [myPlugin]  
12   }  
13 }
```


More examples

- babel-plugin-remove-jquery

AST Explorer

AST Explorer

Snippet

JavaScript

</>

babel7

Transform

default

Parser: babel7-7.0.0-beta.31

Transformer: babel7-7.0.0-beta.31

```
1 /**
2  * Paste or drop some JavaScript here and explore
3  * the syntax tree created by chosen parser.
4  * You can use all the cool new features from ES6
5  * and even more. Enjoy!
6  */
7
8 let tips = [
9   "Click on any AST node with a '+' to expand it",
10
11   "Hovering over a node highlights the \
12    corresponding part in the source code",
13
14   "Shift click on an AST node expands the whole subtree"
15 ];
16
17 function printTips() {
18   tips.forEach((tip, i) => console.log(`Tip ${i}:` + tip));
19 }
20
```

Tree

JSON

AST Explorer

Hide methods

Hide empty keys

Hide location data

Hide type keys

File {

type: "File"

start: 0

end: 476

+ loc: {start, end}

- program: Program {

type: "Program"

start: 0

end: 476

+ loc: {start, end}

sourceType: "module"

- body: [

Previous

```
1 export default function (babel) {
2   const { types: t } = babel;
3
4   return {
5     name: "ast-transform", // not required
6     visitors: {
7       Identifier(path) {
8         path.node.name = path.node.name.split('').reverse().join('');
9       }
10     }
11   };
12 }
13
```

```
1 /**
2  * Paste or drop some JavaScript here and explore
3  * the syntax tree created by chosen parser.
4  * You can use all the cool new features from ES6
5  * and even more. Enjoy!
6  */
7
8 let spit = [
9   "Click on any AST node with a '+' to expand it",
10
11   "Hovering over a node highlights the \
12    corresponding part in the source code",
13
14   "Shift click on an AST node expands the whole subtree"
15 ];
16
17 function spitTnirp() {
18   spit.forEach((tip, i) => console.log(`Tip ${i}:` + spit[i]));
19 }
20
```

Build with: React, Babel, Font Awesome, CodeMirror, Express, and WebPack | GitHub

astexplorer.net