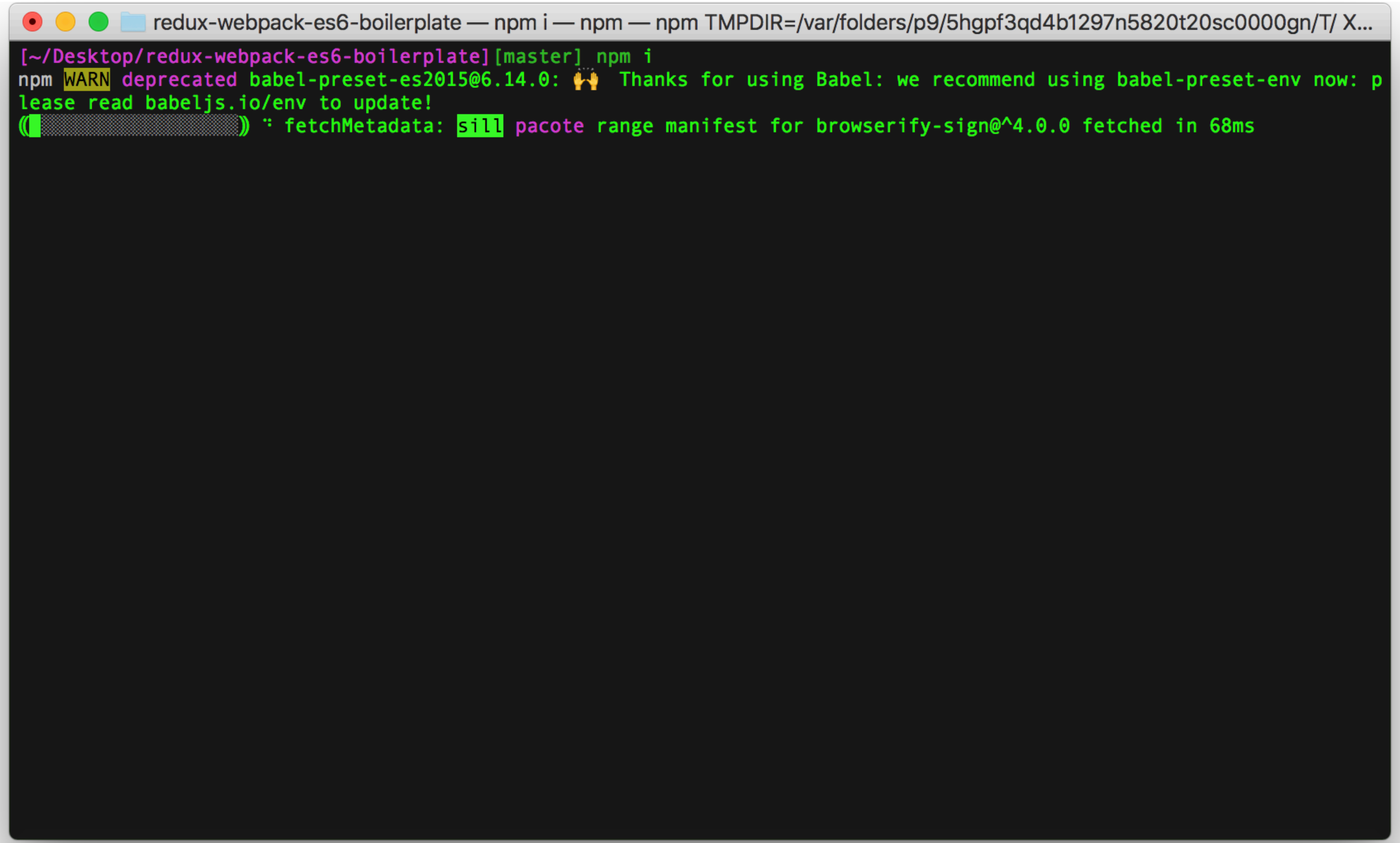
Writing Your First Babel Plugin 🛠

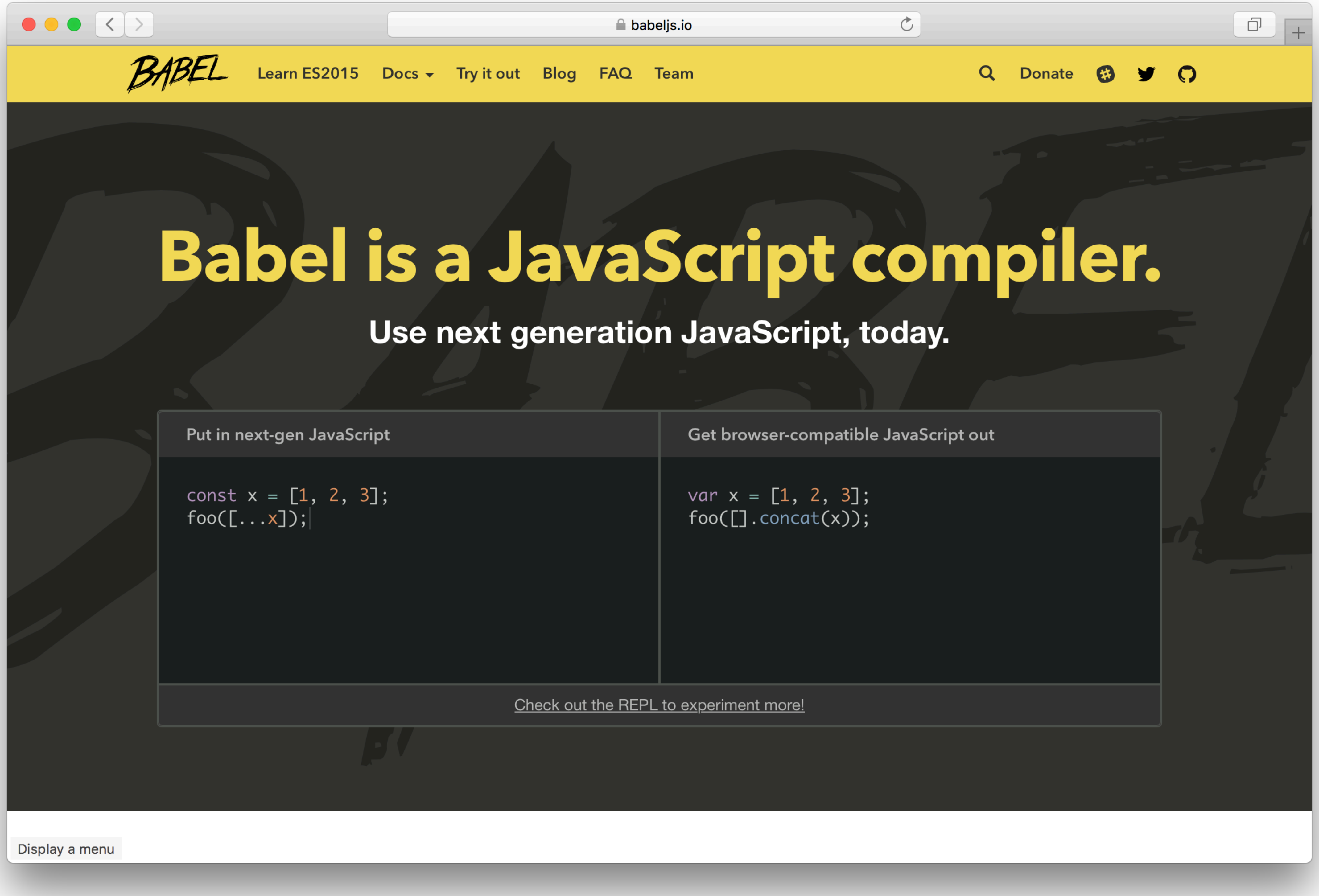
Josh Parnham

What is Babel?



redux-webpack-es6-boilerplate — npm i — npm — npm TMPDIR=/var/folders/p9/5hgpf3qd4b1297n5820t20sc0000gn/T/ X...

```
[~/Desktop/redux-webpack-es6-boilerplate][master] npm i
npm WARN deprecated babel-preset-es2015@6.14.0: 🙌  Thanks for using Babel: we recommend using babel-preset-env now: p
lease read babeljs.io/env to update!
(█░░░░░░░░░░░░░░░░) ⠂ fetchMetadata: sill pacote range manifest for browserify-sign@^4.0.0 fetched in 68ms
```



Babel is a JavaScript compiler.

Use next generation JavaScript, today.

Put in next-gen JavaScript

```
const x = [1, 2, 3];
foo([...x]);
```

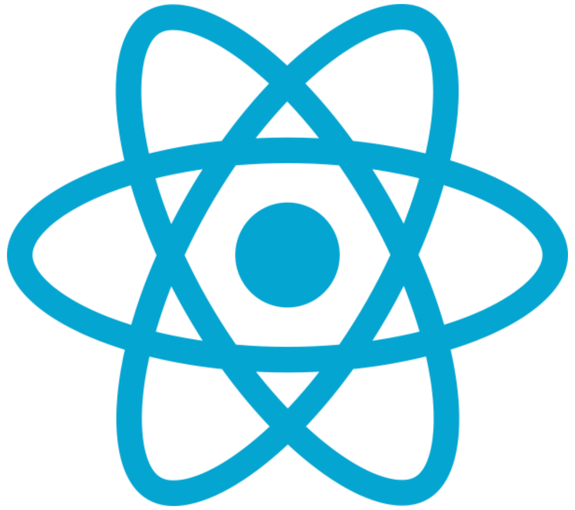
Get browser-compatible JavaScript out

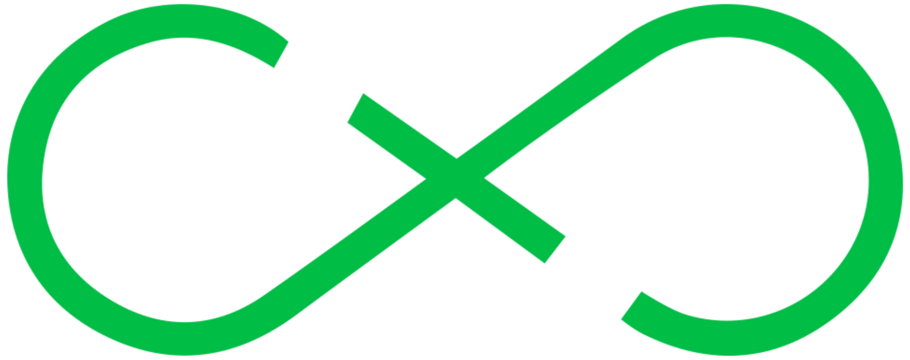
```
var x = [1, 2, 3];
foo([].concat(x));
```

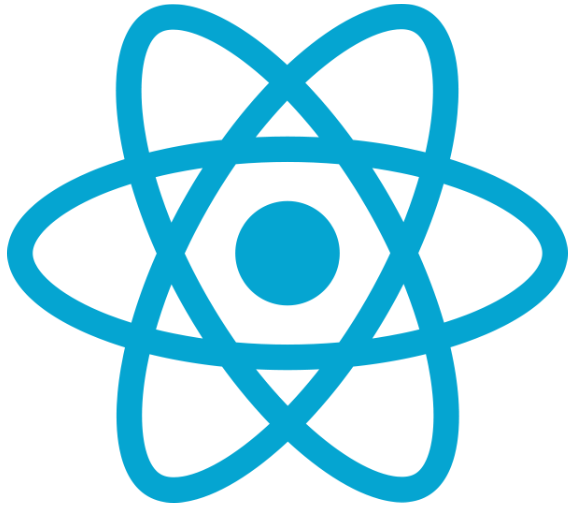
Check out the REPL to experiment more!

How it's used

- Build tools, such as `webpack`
- Via the CLI, `babel-cli`
- Directly through `babel-node`







Configuration

`.babelrc` from `create-react-native-app`:

```
{
  "presets": [
    "babel-preset-react-native-stage-0/decorator-support"
  ],
  "env": {
    "development": {
      "plugins": [
        "transform-react-jsx-source"
      ]
    }
  }
}
```

Configuration

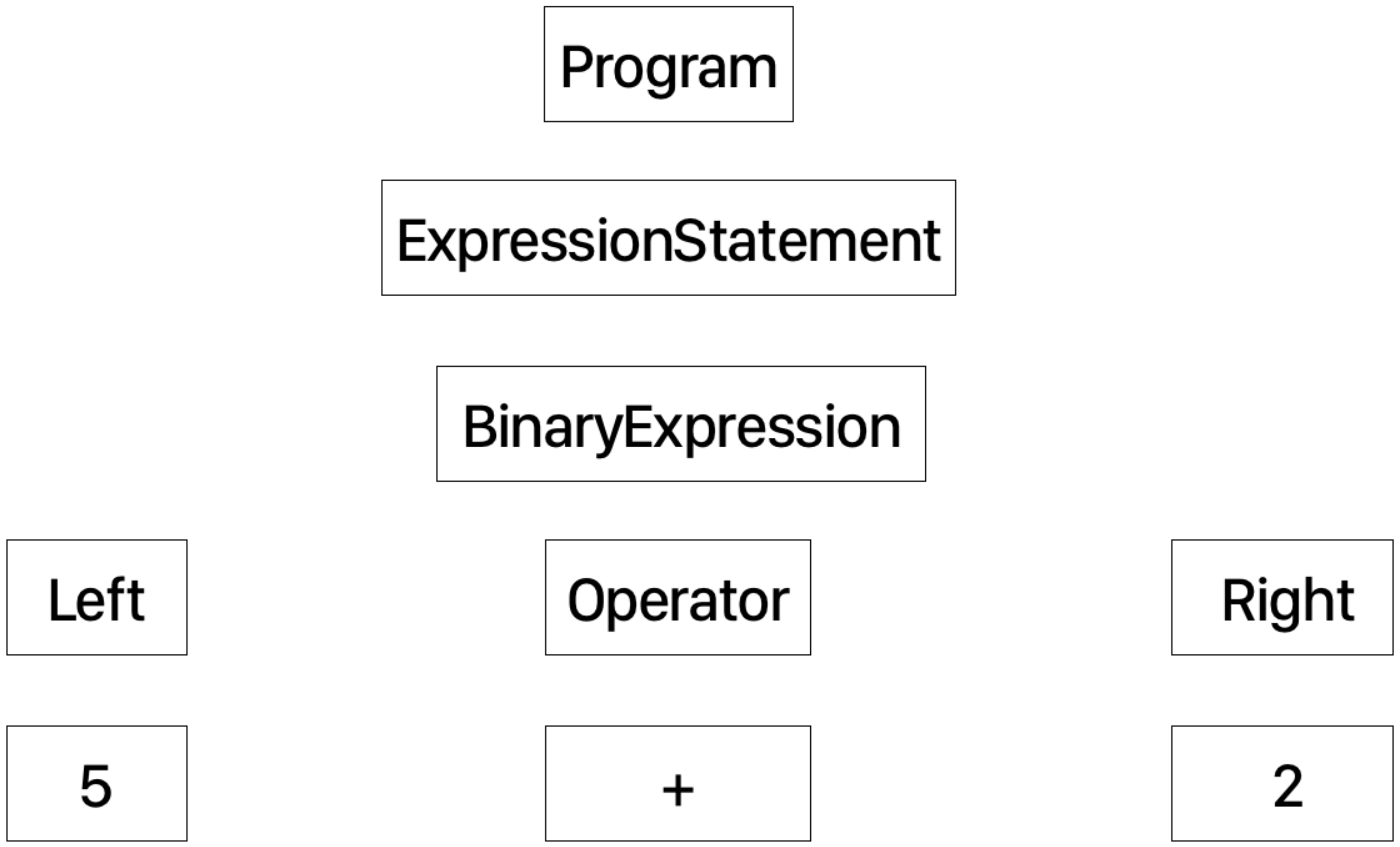
- Preset
  - Set of plugins
- Plugin
  - *Transform & Syntax*

Order of Operation

Parse ➡️ Transform ➡️ Generate

Abstract Syntax Trees (ASTs)

5 + 2



Program

ExpressionStatement

BinaryExpression

Left

Operator

Right

5

+

2

🔗 astexplorer.net

```
5 + 2
```

```
"expression": {
    "type": "BinaryExpression",
    ...
    },
    "left": {
        "type": "Literal",
        "value": 5,
        ...
    },
    "operator": "+",
    "right": {
        "type": "Literal",
        "value": 2,
        ...
    }
}
```

Transformation

- Input & output should be valid (Babylon-supported) JS ✅
  - tc39 stage features
  - Flow, JSX, etc
- Doesn't support third-party syntax plugins ❌

Example

babel-plugin-transform-fun-chaos

```
module.exports = function({ types: t }) {
  return {
    name: 'transform-fun-chaos',
    visitor: {
      BinaryExpression(path) {
        if (path.node.operator !== '+') return

        path.replaceWith(
            t.binaryExpression('-', path.node.left, path.node.right)
        );
      }
    }
  };
};
```

babel-plugin-transform-fun-chaos

```
module.exports = function({ types: t }) {
  return {
    name: 'transform-fun-chaos',
    visitor: {
      BinaryExpression(path) {
        if (path.node.operator !== '+') return

        if (Math.random() > 0.5) {
          path.replaceWith(
            t.binaryExpression('-', path.node.left, path.node.right)
          );
        }
      }
    }
  };
};
```

babel-plugin-transform-fun-chaos

```
module.exports = function({ types: t }) {
  return {
    name: 'transform-fun-chaos',
    visitor: {
      BinaryExpression(path) {
        if (path.node.operator !=='+') return

        const afterFivePM = (new Date()).getHours() >= 17
        if (afterFivePM) {
          path.replaceWith(
            t.binaryExpression('-', path.node.left, path.node.right)
          );
        }
      }
    }
  };
};
```

Actual Example

NaN

```
NaN == NaN        // false
NaN === NaN       // false
Number.isNan(NaN) // true
```

We can write a plugin which automatically converts NaN equality checks to use the correct function

babel-plugin-transform-nan-equality

```
module.exports = function({ types: t }) {
  return {
    name: 'transform-nan-equality',
    visitor: {
      BinaryExpression(path) {
        if (
          (path.node.operator !== '==' && path.node.operator !== '===') &&
          (path.node.left.name !== t.identifier('NaN').name) &&
          (path.node.right.name !== t.identifier('NaN').name)) {
          return
        }

        if (path.node.left.name === t.identifier('NaN').name) {
            path.replaceWithSourceString(`isNan(${path.node.right.name})`)
        } else if (path.node.right.name === t.identifier('NaN').name) {
            path.replaceWithSourceString(`isNan(${path.node.left.name})`)
        }
      }
    }
  };
};
```

Should anyone ever actually do this?

🙈

Should anyone ever actually do this?

Nah.

Thanks ✌️

@joshparnham