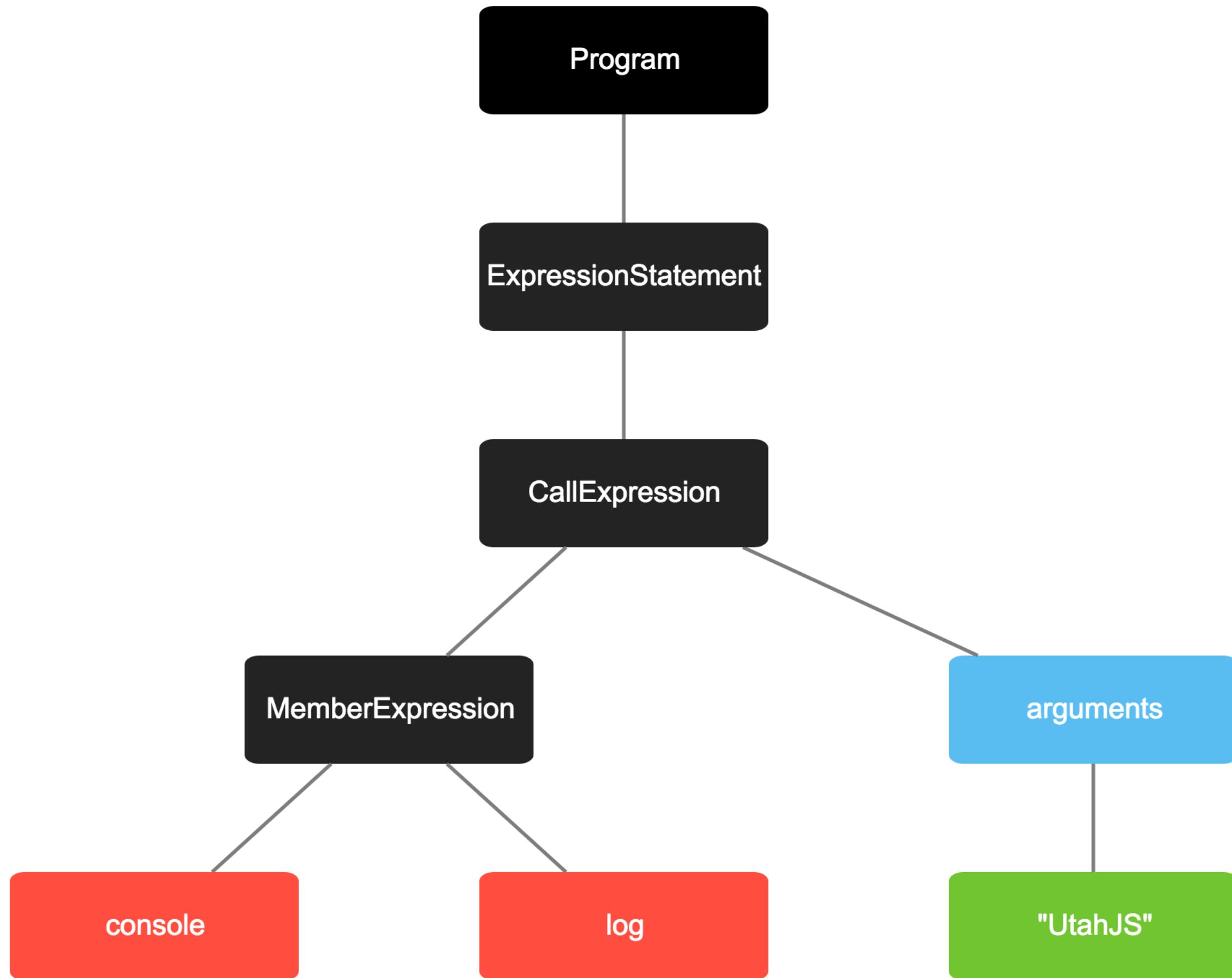


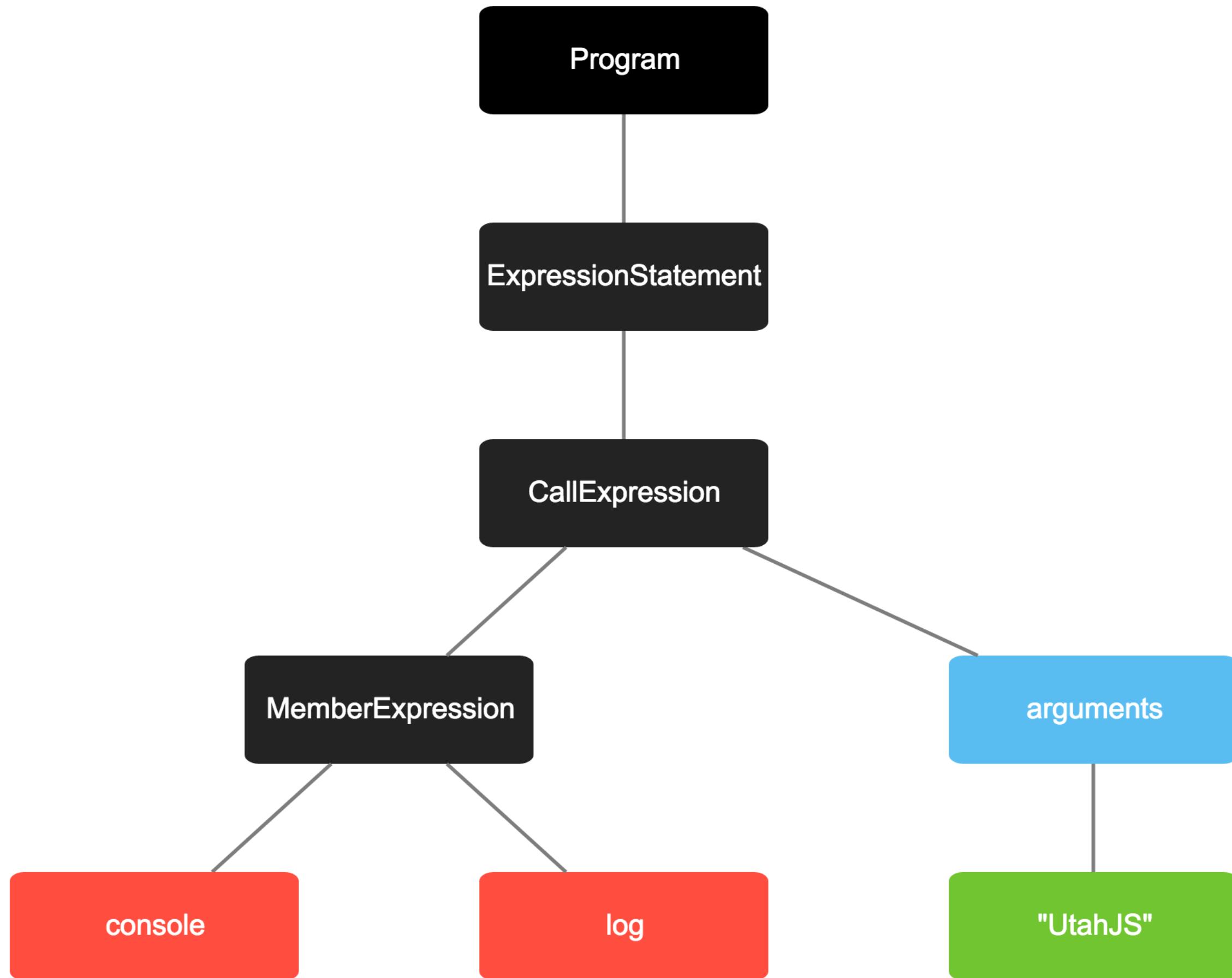
Harnessing the Power of *Abstract Syntax Trees*

Parsons

```
console.log("UtahJS");
```



```
{
  type: "Program",
  body: [
    {
      type: "ExpressionStatement",
      expression: {
        type: "CallExpression",
        callee: {
          type: "MemberExpression",
          computed: false,
          object: {
            type: "Identifier",
            name: "console"
          },
          property: {
            type: "Identifier",
            name: "log"
          }
        },
        arguments: [
          {
            type: "Literal",
            value: "UtahJS",
            raw: "\"UtahJS\""
          }
        ]
      }
    }
  ]
}
```



Acorn

Espresso

PARSING

```
// generate an AST from a string of code  
espree.parse("console.log('UtahJS')");
```

PARSING

```
// generate an AST from a string of code  
acorn.parse("console.log('UtahJS')");
```

PARSING ES6

```
// generate an AST from a string of code  
acorn.parse("console.log('UtahJS')", { ecmaVersion: 6 });
```



A STS ARE
EVERWHERE

How many of you use *babel*?



sebmck authored on Jul 1

latest commit 990b1f37ba

README.md

first commit

5 months ago

index.js

update example to use 5.6.0+ api

2 months ago

package.json

first commit

5 months ago

README.md

babel-plugin-example

Bob **hates** function declarations with a passion (nobody knows why). Bob loves to enforce this on his coworkers, he's snuck this plugin into their build system to force his tyrannical code style.

Usage

BABEL PLUGIN

```
module.exports = function (Babel) {
  return new Babel.Plugin("plugin-example", {
    // visitor: {
    //   "FunctionDeclaration": swapWithExpression
    // }
  } );
};
```

BABEL PLUGIN

```
module.exports = function (Babel) {
  return new Babel.Plugin("plugin-example", {
    visitor: {
      "FunctionDeclaration": swapWithExpression
    }
  });
};
```

GUTS

```
function swapWithExpression(node, parent) {  
  var id = node.id;  
  // change the node type  
  // node.type = "FunctionExpression";  
  // node.id  = null;  
  // return a variable declaration  
  // return Babel.types.variableDeclaration("var", [  
    //   Babel.types.variableDeclarator(id, node)  
    // ]);  
}  
}
```

GUTS

```
function swapWithExpression(node, parent) {  
  var id = node.id;  
  // change the node type  
  node.type = "FunctionExpression";  
  node.id  = null;  
  // return a variable declaration  
  // return Babel.types.variableDeclaration("var", [  
    //   Babel.types.variableDeclarator(id, node)  
    // ]);  
}  
}
```

GUTS

```
function swapWithExpression(node, parent) {  
  var id = node.id;  
  // change the node type  
  node.type = "FunctionExpression";  
  node.id  = null;  
  // return a variable declaration  
  return Babel.types.variableDeclaration("var", [  
    Babel.types.variableDeclarator(id, node)  
  ]);  
}
```

**KNOW YOUR
NODES**

ES6 SPECIFICS

STATEMENTS

- ForOfStatement

EXPRESSIONS

- ArrowFunctionExpression
- YieldExpression

TEMPLATE LITERALS

- TemplateLiteral
- TaggedTemplateExpression
- TemplateElement

PATTERNS

- ObjectPattern
- ArrayPattern
- RestElement
- AssignmentPattern

CLASSES

- ClassBody
- MethodDefinition
- ClassDeclaration
- ClassExpression
- MetaProperty

MODULES

- ModuleDeclaration
- ModuleSpecifier
- ImportDeclaration
- ImportSpecifier
- ImportDefaultSpecifier
- ImportNamespaceSpecifier
- ExportNamedDeclaration
- ExportSpecifier
- ExportDefaultDeclaration
- ExportAllDeclaration



estree / estree

Watch ▾ 63



The ESTree Spec

112 commits

1 branch

0 releases

16 contributors



Branch: master ▾

estree / +



Fix TemplateElement.value ...

 RReverser authored on Jul 30

latest commit bc81a9a890 

 experimental move experimental documents into a separate folder 6 months ago

 README.md Add philosophy section to README 6 months ago

 deprecated.md finished migrating all spec text 7 months ago

 es6.md Fix TemplateElement.value 2 months ago

 spec.md Remove deprecated guard property 2 months ago

 README.md

The ESTree Spec

Once upon a time, an [unsuspecting Mozilla engineer](#) created an API in Firefox that exposed the SpiderMonkey engine's JavaScript parser as a JavaScript API. Said engineer [documented the format it produced](#), and this format caught on as a lingua franca for tools that manipulate JavaScript source

**ANAST GIVES
YOU SUPER
POWERS**

*Making Easy Things Easy
& Hard Things Possible*

6th Edition
Covers Perl 5.14

Learning Perl



O'REILLY®

Randal L. Schwartz,
brian d foy & Tom Phoenix

*"Making easy things easy & hard
things possible"*

- Learning Perl

JS AWARE

git diff

```
console.log("UtahJS");
```

```
console.log("SomeOtherConf");
```

```
console.error("SomeOtherConf");
```

```
~/Dropbox/Talks/utahjs 2015 (master) $ git diff
diff --git a/tree1.js b/tree1.js
index 9a0b08f..b547a58 100644
--- a/tree1.js
+++ b/tree1.js
@@ -1 +1 @@
-console.log("UtahJS");
+console.error("SomeOtherConf");
~/Dropbox/Talks/utahjs 2015 (master) $ █
```

yes we can!

Version 1

1. Create ASTs from the old and new files
2. Run a tree-diffing algorithm
3. Display the differences in a useful way

git diff

```
$ git diff --raw  
:100644 100644 9a0b08f... 0000000... M tree1.js
```

```
$ git diff --raw  
:100644 100644 9a0b08f... 0000000... M tree1.js
```

File Permissions	MD5 Hash	MD5 Hash	Status	Filename
-------------------------	-----------------	-----------------	---------------	-----------------

```
git diff --raw | node compare.js
```

```
// let's read all of this input from stdin into an array
var lines = fs.readFileSync('/dev/stdin').toString().split('\n');
```

```
// lines now looks something like this  
[ ':100644 100644 9a0b08f... 0000000... M tree1.js' ]
```

```
lines.map(function(line) {  
    var parts = line.split(' ');  
    // var file = parts.pop().split('\t');  
    // return [file[1], parts[2].slice(0, -3)];  
});
```

```
lines.map(function(line) {  
    var parts = line.split(' ');\n    var file = parts.pop().split('\\t');\n    // return [file[1], parts[2].slice(0, -3)];\n});
```

```
lines.map(function(line) {  
    var parts = line.split(' ');\n    var file = parts.pop().split('\\t');\n    return [file[1], parts[2].slice(0, -3)];\n});
```

```
// the key parts of our git diff  
[ [ "tree1.js", "9a0b08f" ] ]
```

PARSING

```
// generate an AST from a string of code  
espree.parse("console.log('UtahJS')");
```

```
.map(function(files) {
  var after = fs.readFileSync(files[0]);
  // var before = child_process.execSync("git show" + files[1]);
  // return {
  //   filename: files[0],
  //   before: espree.parse(before, options),
  //   after: espree.parse(after, options)
  // };
})
```

```
.map(function(files) {
  var after = fs.readFileSync(files[0]);
  var before = child_process.execSync("git show" + files[1]);
  // return {
  //   filename: files[0],
  //   before: espree.parse(before, options),
  //   after: espree.parse(after, options)
  // };
})
```

```
.map(function(files) {
  var after = fs.readFileSync(files[0]);
  var before = child_process.execSync("git show" + files[1]);
  return {
    filename: files[0],
    before: espree.parse(before, options),
    after: espree.parse(after, options)
  };
})

```

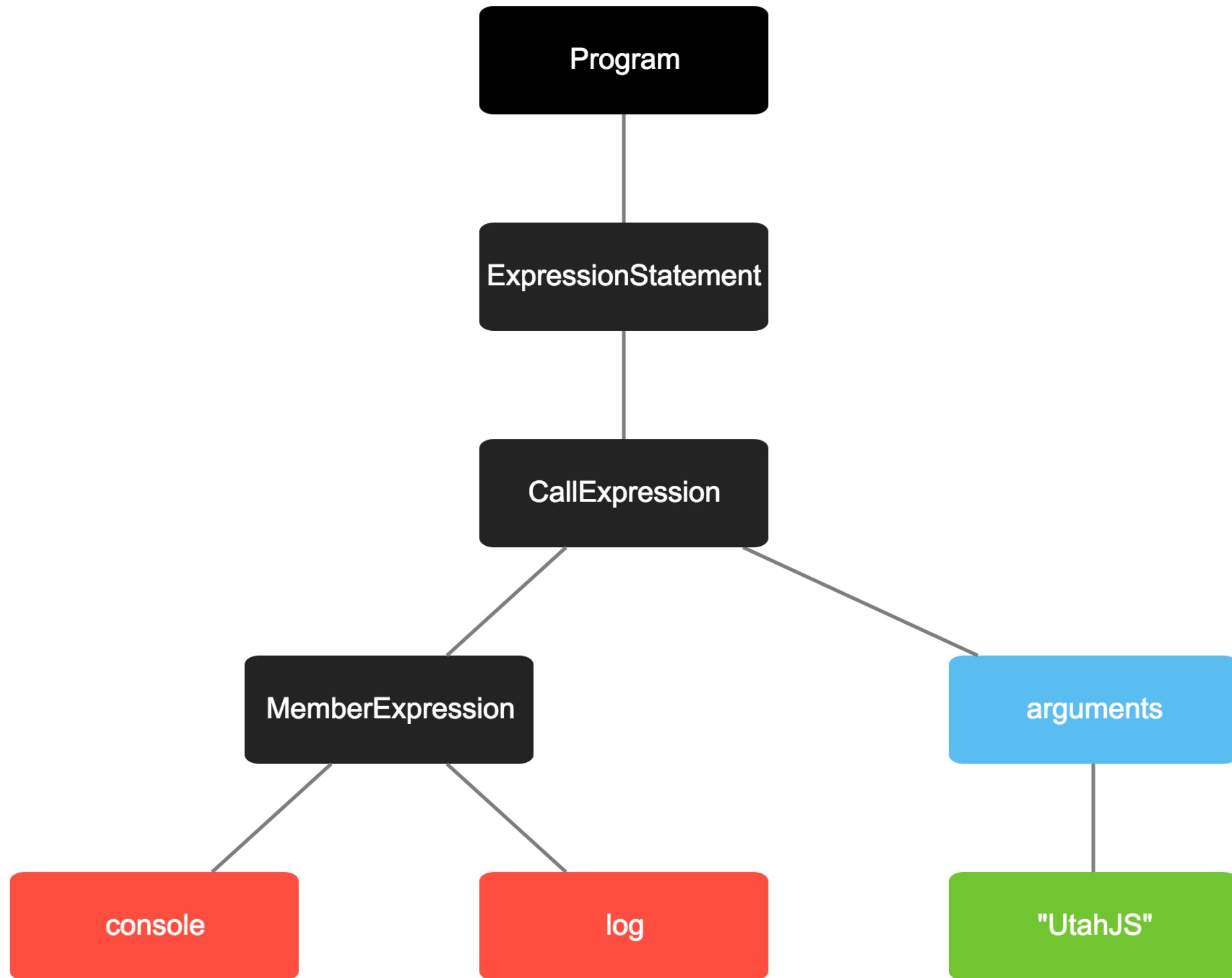
```
.map(function(files) {
  var after = fs.readFileSync(files[0]);
  var before = child_process.execSync("git show" + files[1]);
  return {
    filename: files[0],
    before: espree.parse(before, options),
    after: espree.parse(after, options)
  };
})

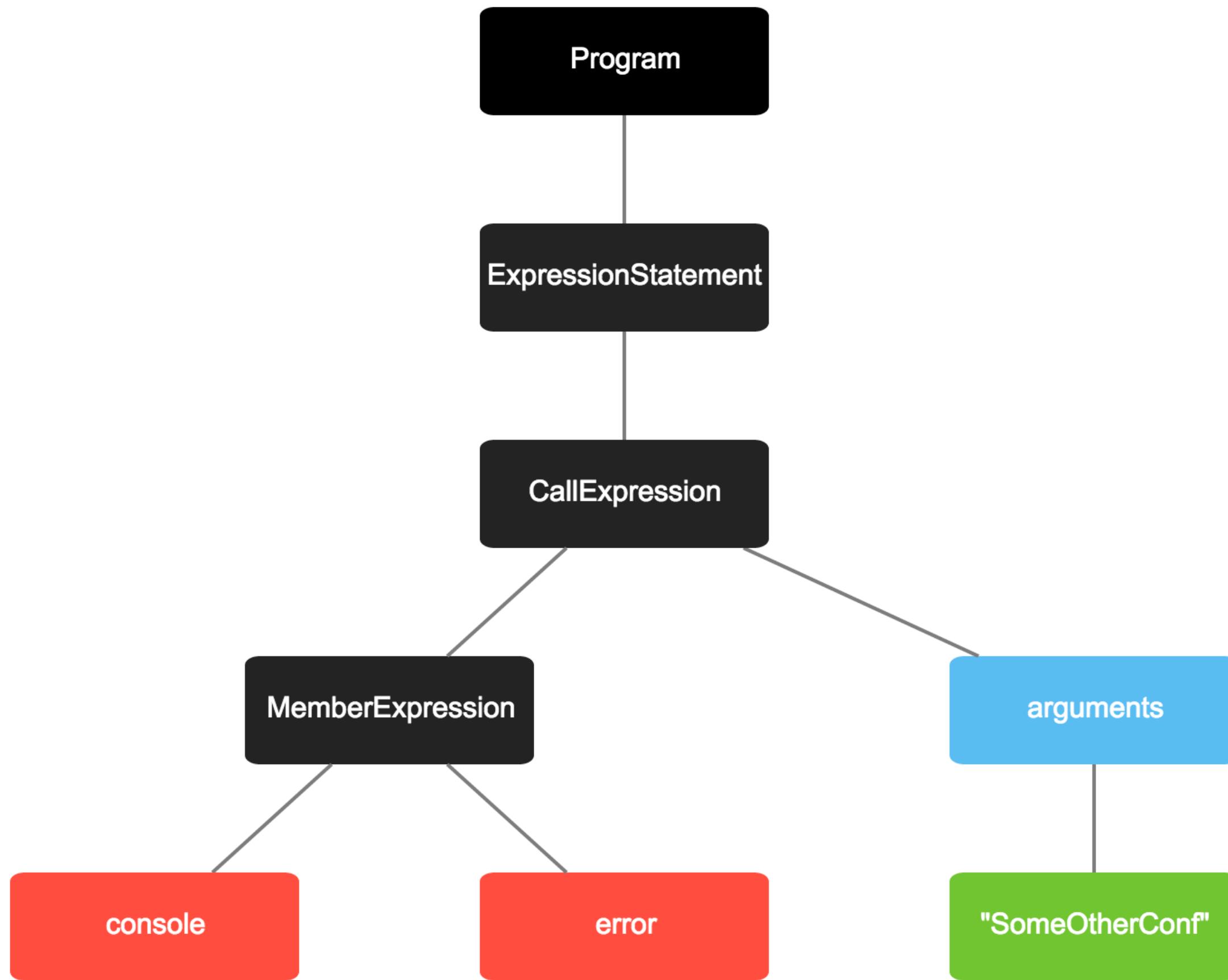
```

```
[ {  
    filename: "trees1.js",  
    before: { type: "Program", body: [Object] },  
    after: { type: "Program", body: [Object] }  
}]
```

FITS ON ONE SLIDE*

```
var lines = fs.readFileSync('/dev/stdin').toString().split('\n');
var trees = lines.map(function(line) {
  var parts = line.split(' ');
  var file = parts.pop().split('\t');
  return [path.resolve(file[1]), parts[2].slice(0, -3)];
}).filter(function(files) {
  return files[0].indexOf('.js') > -1;
}).map(function(files) {
  var after = fs.readFileSync(files[0]);
  var before = child_process.execSync("git show " + files[1]);
  return {
    filename: files[0],
    before: espree.parse(before, options),
    after: espree.parse(after, options)
  };
});
```





Step 2

```
// let's see if something changed  
var different = deepEqual(treeBefore, treeAfter);
```

```
function deepEqual(a, b) {  
    if (a === b) {  
        return true;  
    }  
    if (!a || !b) {  
        return false;  
    }  
    if (Array.isArray(a)) {  
        return a.every(function(item, i) {  
            return deepEqual(item, b[i]);  
        });  
    }  
    if (typeof a === 'object') {  
        return Object.keys(a).every(function(key) {  
            return deepEqual(a[key], b[key]);  
        });  
    }  
    return false;  
}
```

```
function deepEqual(a, b) {  
    // if (a === b) {  
    //     return true;  
    // }  
    // if (!a || !b) {  
    //     return false;  
    // }  
    // if (Array.isArray(a)) {  
    //     return a.every(function(item, i) {  
    //         return deepEqual(item, b[i]);  
    //     });  
    // }  
    if (typeof a === 'object') {  
        return Object.keys(a).every(function(key) {  
            return deepEqual(a[key], b[key]);  
        });  
    }  
    return false;  
}
```

```
if (typeof a === 'object') {
  return Object.keys(a).every(function(key) {
    return deepEqual(a[key], b[key]);
  });
}
return false;
```

```
if (typeof a === 'object') {
  var equal = Object.keys(a).every(function(key) {
    return deepEqual(a[key], b[key]);
  });
  return equal;
}
return false;
```

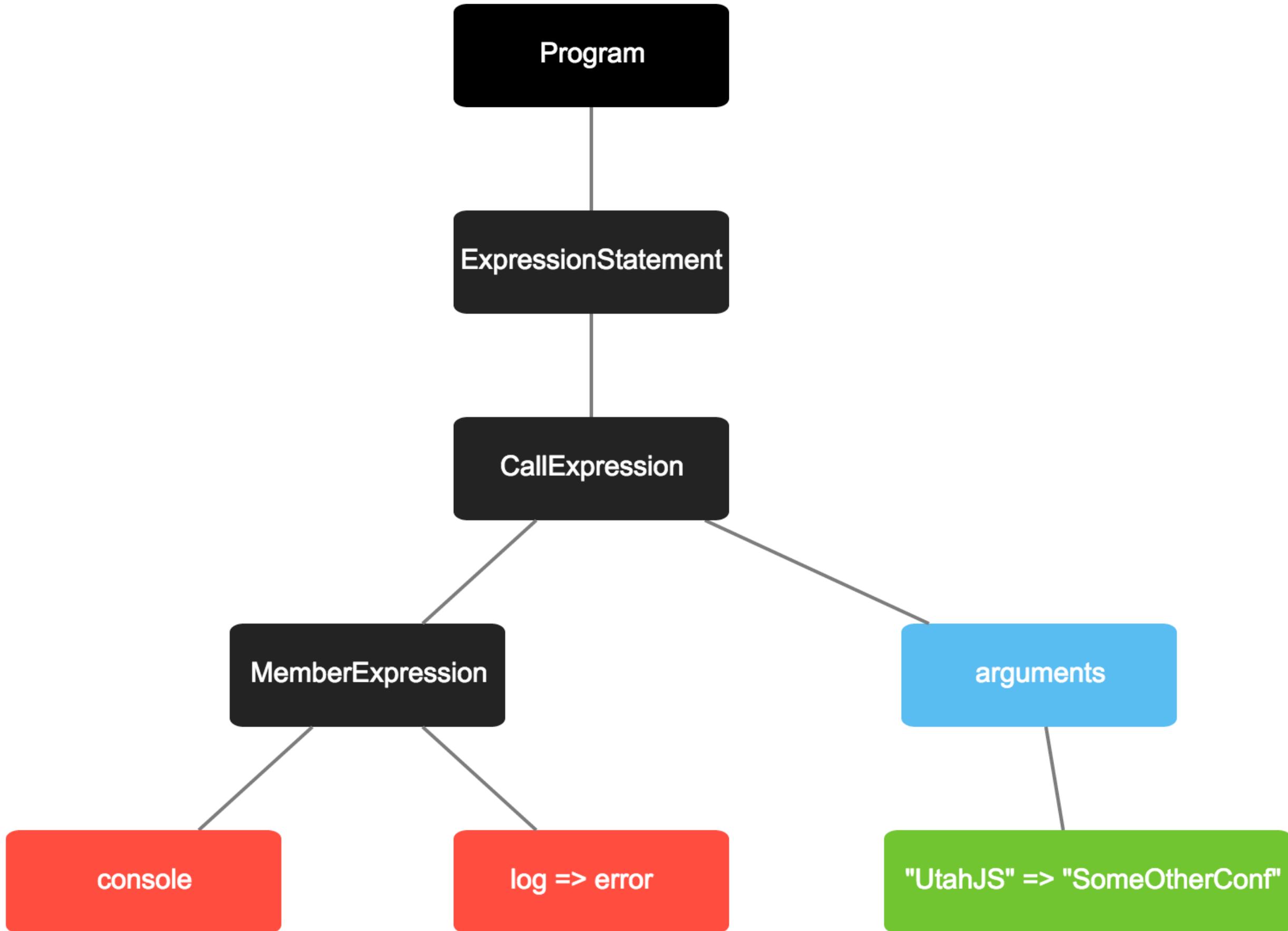
```
if (typeof a === 'object') {
    // var equal = Object.keys(a).every(function(key) {
    //     return deepEqual(a[key], b[key]);
    // });
    if (!equal) {
        console.log('[' + a.type + '] => [' + b.type + ']');
    }
    // return equal;
}

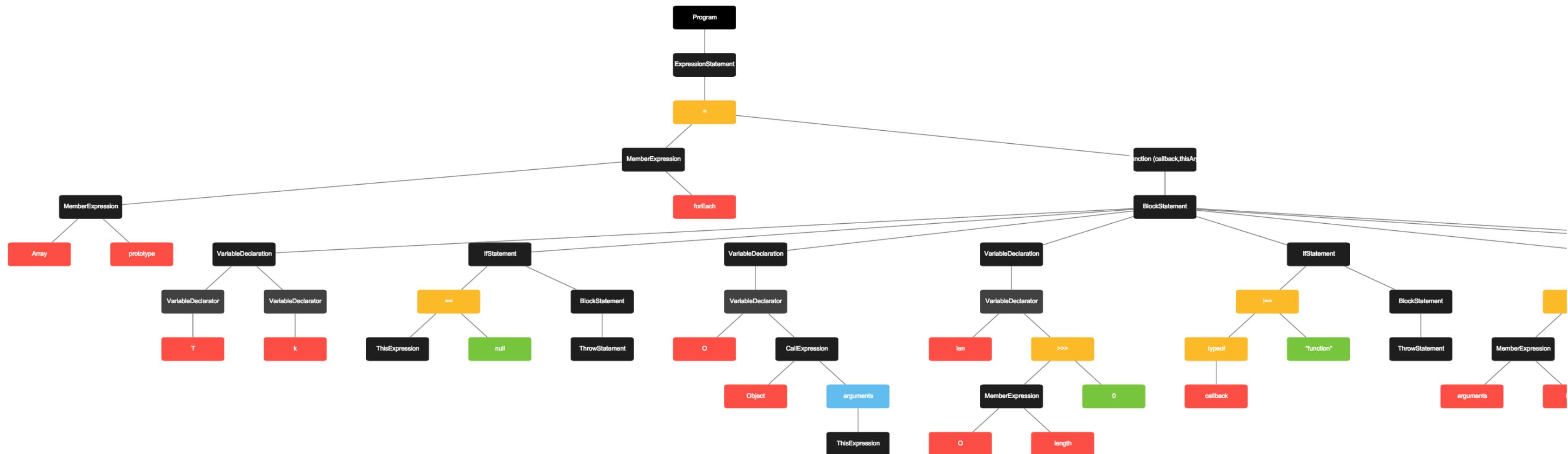
console.log('"' + a + '"' => '"' + b + '"');
// return false;
```

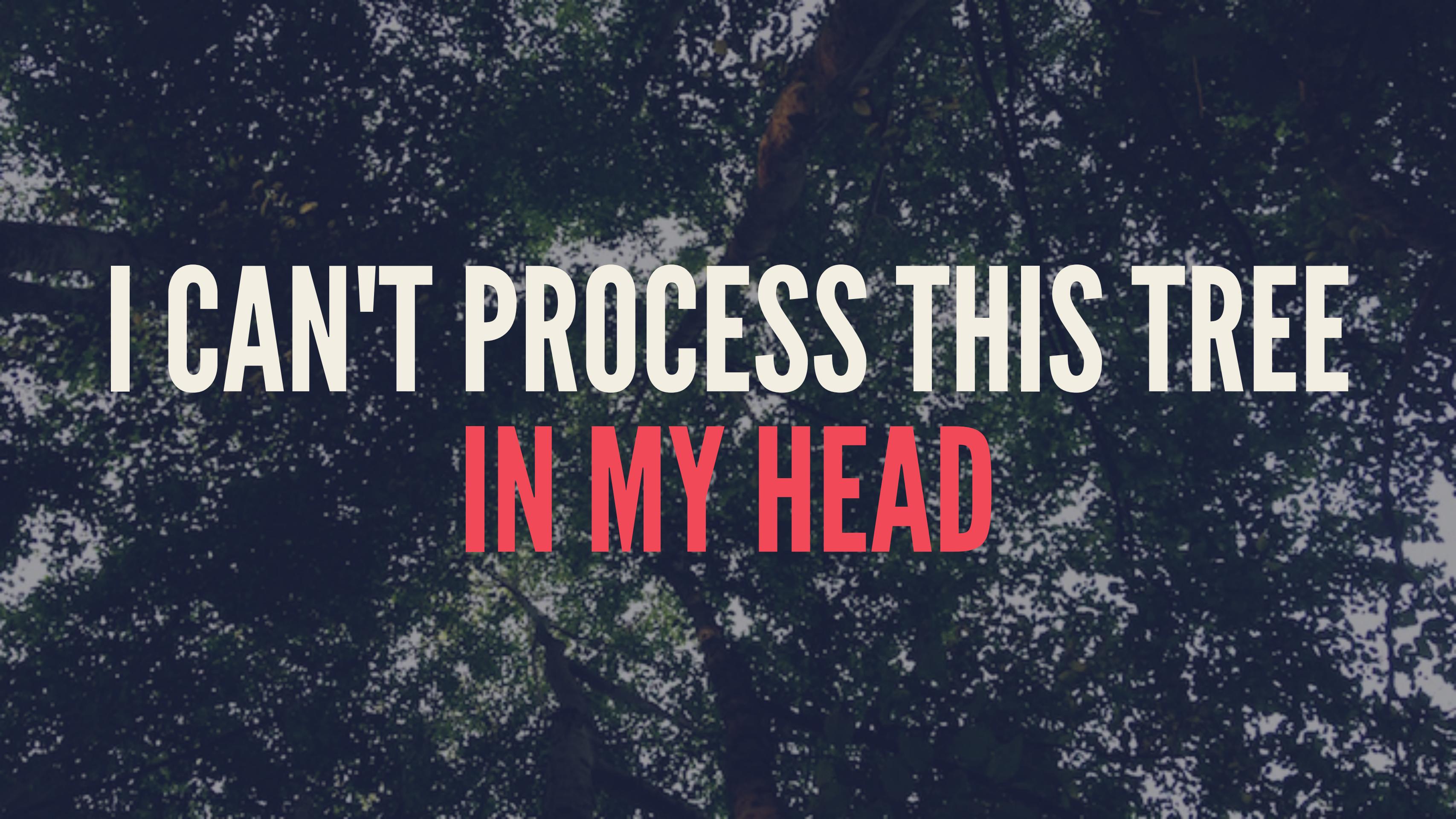
```
git diff --raw | node compare.js

"log" => "error"
[Identifier] => [Identifier]
[MemberExpression] => [MemberExpression]
[CallExpression] => [CallExpression]
[ExpressionStatement] => [ExpressionStatement]
[Program] => [Program]
```

```
~/Dropbox/Talks/utahjs 2015 (master) $ git diff
diff --git a/tree1.js b/tree1.js
index 9a0b08f..b547a58 100644
--- a/tree1.js
+++ b/tree1.js
@@ -1 +1 @@
-console.log("UtahJS");
+console.error("SomeOtherConf");
~/Dropbox/Talks/utahjs 2015 (master) $ █
```





A close-up photograph of a tree trunk. The bark is dark brown with prominent, irregular white lichen patches and some small, thin brown roots hanging down from the side.

I CAN'T PROCESS THIS TREE
IN MY HEAD

TOWARD A BETTER DIFF

- require() statement changes?
- Changes to variables?
- Function argument changes
- Breaking Changes?

```
export function buildHouse(lot, color, size, bedrooms) {  
    clearLot(lot);  
    let foundation = buildFoundation(size);  
    let walls = buildWalls(bedrooms);  
    let paintedWalls = paintWalls(color, walls);  
    let roof = buildRoof(foundation, walls);  
    let house = foundation + paintedWalls + roof;  
  
    // house is all done right-away  
    return house;  
}
```

```
function getPermits(callback) {
  setTimeout(callback, 1.0519e10); // 4 months because trees
}

export function buildHouse(lot, color, size, bedrooms, callback) {
  getPermits((permits) => {
    clearLot(permits, lot);
    let foundation = buildFoundation(size);
    let walls = buildWalls(bedrooms);
    let paintedWalls = paintWalls(color, walls);
    let roof = buildRoof(foundation, walls);
    let house = foundation + paintedWalls + roof;

    // house will be ready in about a year
    callback(house);
  });
}
```

```
~/Dropbox/Talks/utahjs 2015 (master) $ git diff -w complex1.js
diff --git a/complex1.js b/complex1.js
index 12ea4be..ec5d1ce 100644
--- a/complex1.js
+++ b/complex1.js
@@ -1,11 +1,17 @@
-export function buildHouse(lot, color, size, bedrooms) {
-    clearLot(lot);
+function getPermits(callback) {
+    setTimeout(callback, 1.0519e10); // 4 months because trees
+}
+
+export function buildHouse(lot, color, size, bedrooms, callback) {
+    getPermits((permits) => {
+        clearLot(permits, lot);
+        let foundation = buildFoundation(size);
+        let walls = buildwalls(bedrooms);
+        let paintedwalls = paintwalls(color, walls);
+        let roof = buildRoof(foundation, walls);
+        let house = foundation + paintedwalls + roof;

-        // house is all done right-away
-        return house;
+        // house will be ready in a year
+        callback(house);
+    });
}
```

OUR GOAL

```
git diff --raw | node compare.js
```

house.js

1. The exported `buildHouse` function output went from a return to a callback.
2. The private `getPermits` function was added.

OUR DATA STRUCTURE

```
[{  
    filename: "trees1.js",  
    before: { type: "Program", body: [Object] },  
    after: { type: "Program", body: [Object] }  
}]
```

INTRODUCING ESRECURSE

```
var esrecurse = require('esrecurse');

esrecurse.visit(ast, {
  FunctionDeclaration: function(node) {
    console.log(node);
  }
});
```

VISITING OUR TREES

```
esrecurse.visit(diff.before, {  
  
    // export function a() {}  
    ExportNamedDeclaration: function(node) {  
        var details = inspectFunction(node.declaration, "exported");  
        functions[details.name].before = details;  
    },  
  
    // function a() {}  
    // FunctionDeclaration: function(node) {  
    //     var details = inspectFunction(node.declaration);  
    //     functions[details.name].before = details;  
    // }  
});
```

VISITING OUR TREES

```
esrecurse.visit(diff.before, {  
  
    // export function a() {}  
    ExportNamedDeclaration: function(node) {  
        var details = inspectFunction(node.declaration, "exported");  
        functions[details.name].before = details;  
    },  
  
    // function a() {}  
    FunctionDeclaration: function(node) {  
        var details = inspectFunction(node.declaration);  
        functions[details.name].before = details;  
    }  
});
```

INSPECTING FUNCTION DECLARATIONS

```
function inspectFunction(node, visibility) {  
  return {  
    name: node.id.name, // "buildHouse",  
    // params: node.params.map(function(param) {  
    //   return param.name;  
    // }), // ["lot", "color", "size", ...]  
    // visibility: visibility || "private",  
    // outputType: getOutputType(node)  
  };  
}
```

INSPECTING FUNCTION DECLARATIONS

```
function inspectFunction(node, visibility) {  
  return {  
    name: node.id.name, // "buildHouse"  
    params: node.params.map(function(param) {  
      return param.name;  
    }), // ["lot", "color", "size", ...]  
    // visibility: visibility || "private",  
    // outputType: getOutputType(node)  
  };  
}
```

INSPECTING FUNCTION DECLARATIONS

```
function inspectFunction(node, visibility) {  
  return {  
    name: node.id.name, // "buildHouse"  
    params: node.params.map(function(param) {  
      return param.name;  
    }), // ["lot", "color", "size", ...]  
    visibility: visibility || "private",  
    // outputType: getOutputType(node)  
  };  
}
```

INSPECTING FUNCTION DECLARATIONS

```
function inspectFunction(node, visibility) {  
  return {  
    name: node.id.name, // "buildHouse"  
    params: node.params.map(function(param) {  
      return param.name;  
    }), // ["lot", "color", "size", ...]  
    visibility: visibility || "private",  
    outputType: getOutputType(node)  
  };  
}
```

```
git diff --raw | node compare.js
```

```
[ filename: "house.js",
  functions: {
    buildHouse: {
      before: { name: "buildHouse", /* ... */ },
      after: { name: "buildHouse", /* ... */ }
    }
    getPermits: {
      after: { name: "getPermits", /* ... */ }
    }
  }
}
```

```
{  
  name: "getPermits",  
  visibility: "private",  
  params: [ "callback" ],  
  outputType: "callback"  
}
```

Data => Words

UNROLLING OUR ARRAY

```
.map(function(diff) {  
  return diff.filename + "\n" + getReadableOutput(diff.functions);  
}).join("\n");
```

MEANINGFUL DATA

```
function getReadableOutput(functions) {
  return Object.keys(functions).reduce(function(prev, curr, i) {
    // var name = curr;
    // var visibility = functions[name].after.visibility;
    // var whatHappened = getWhatHappened(functions[name]);
    // return prev + `${i + 1}. The ${visibility} ${name} function ${whatHappened}.\n`;
  }, "");
}
```

MEANINGFUL DATA

```
function getReadableOutput(functions) {
  return Object.keys(functions).reduce(function(prev, curr, i) {
    var name = curr;
    var visibility = functions[name].after.visibility;
    var whatHappened = getWhatHappened(functions[name]);
    // return prev + `${i + 1}. The ${visibility} ${name} function ${whatHappened}.\n`;
  }, "");
}
```

MEANINGFUL DATA

```
function getReadableOutput(functions) {  
  return Object.keys(functions).reduce(function(prev, curr, i) {  
    var name = curr;  
    var visibility = functions[name].after.visibility;  
    var whatHappened = getWhatHappened(functions[name]);  
    return prev + `${i + 1}. The ${visibility} ${name} function ${whatHappened}.\n`;  
  }, "");  
}
```

HUMAN READABLE FTW!

```
function getWhatHappened(func) {  
  
  if (!func.before) {  
    return "was added"  
  }  
  if (!func.after) {  
    return "was removed"  
  }  
  if (func.before.outputType !== func.after.outputType) {  
    return "output went from a " + func.before.outputType + " to a " + func.after.outputType;  
  }  
}
```

A HAPPY ENDING

```
git diff --raw | node compare.js
```

house.js

1. The exported `buildHouse` function output went from a return to a callback.
2. The private `getPermits` function was added.



TREES ARE SUPER
POWERFUL



FINE CINEMA
Warner Company



QUESTIONS?



INE CINEMA
Warner Company

