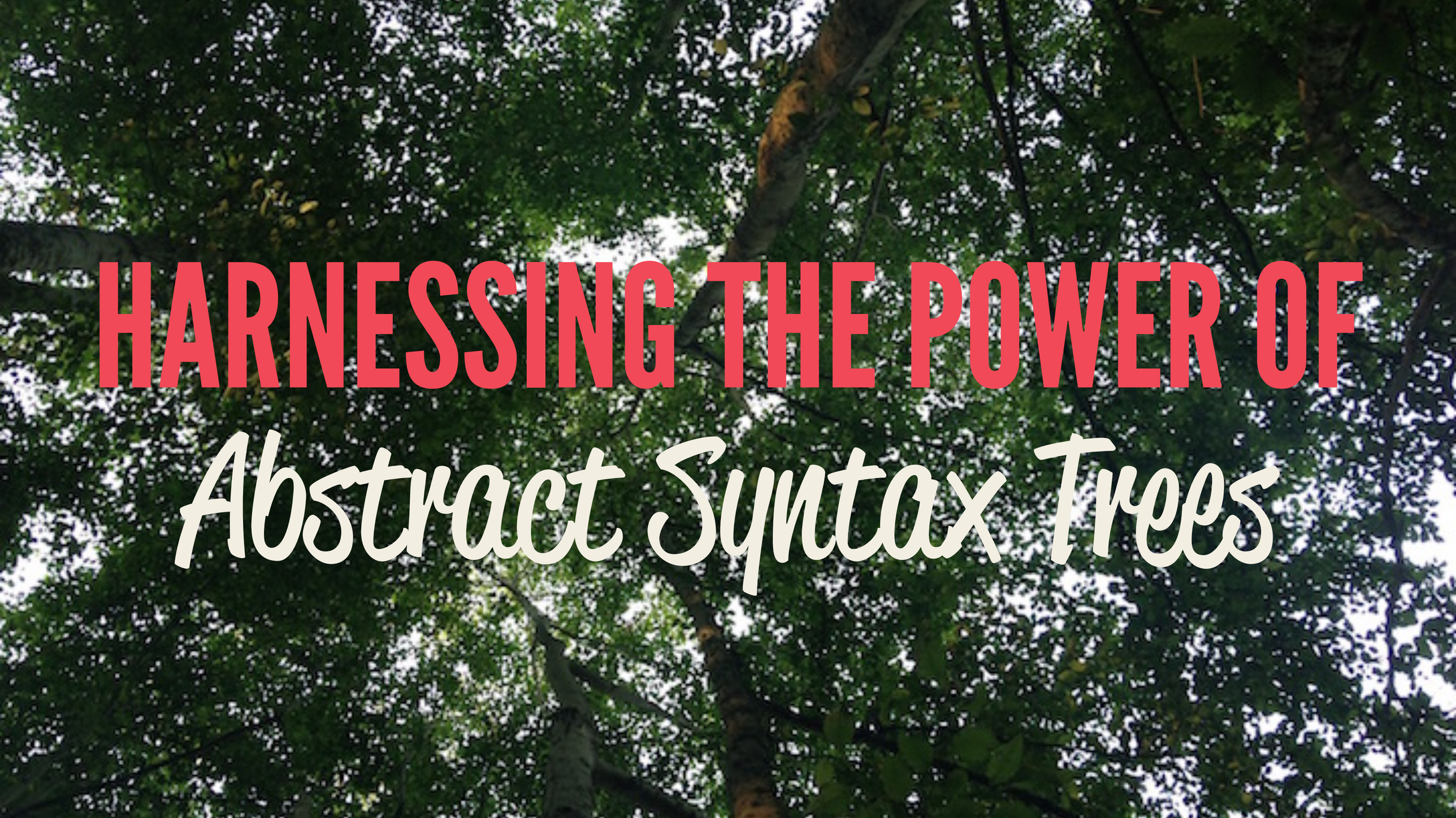
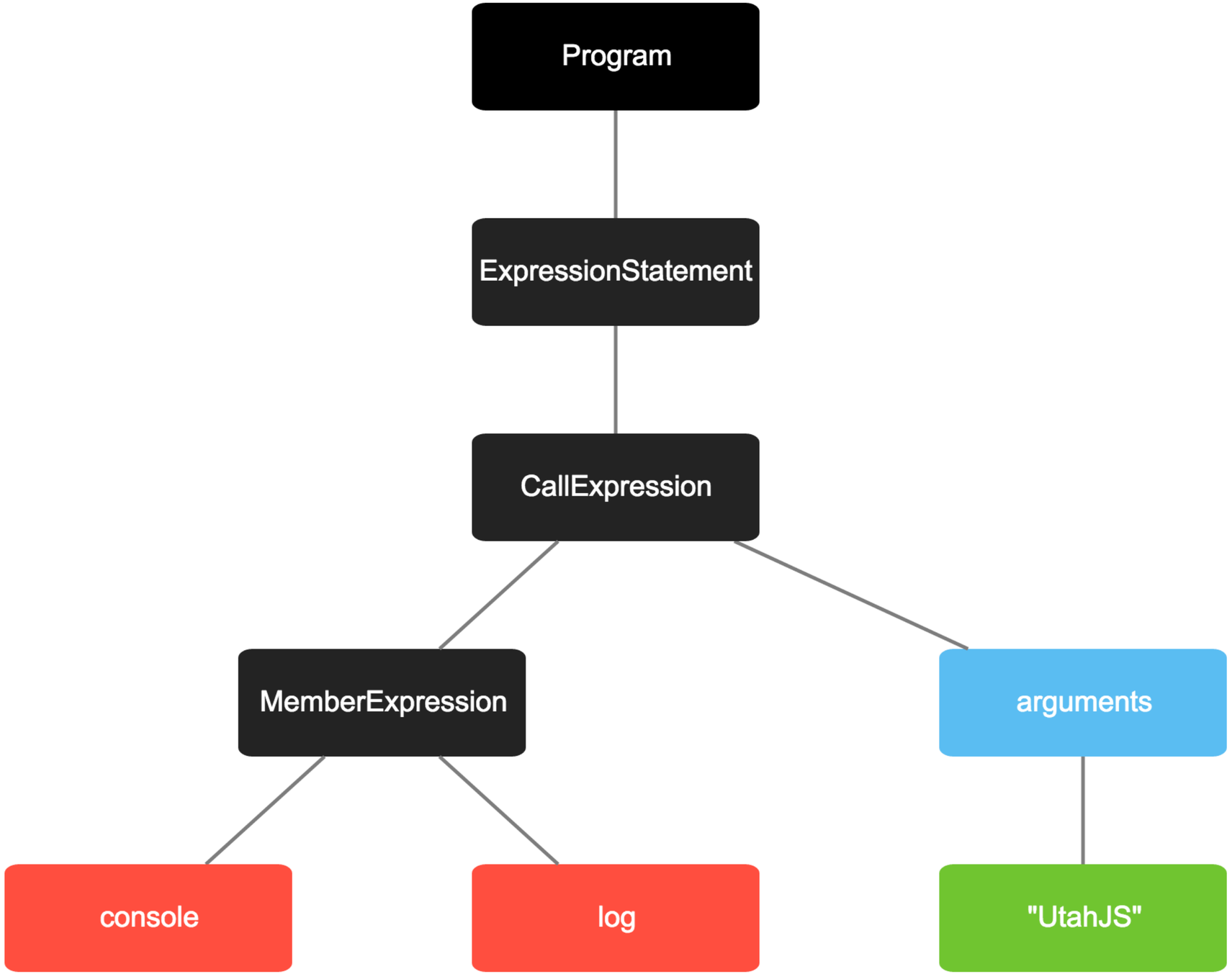
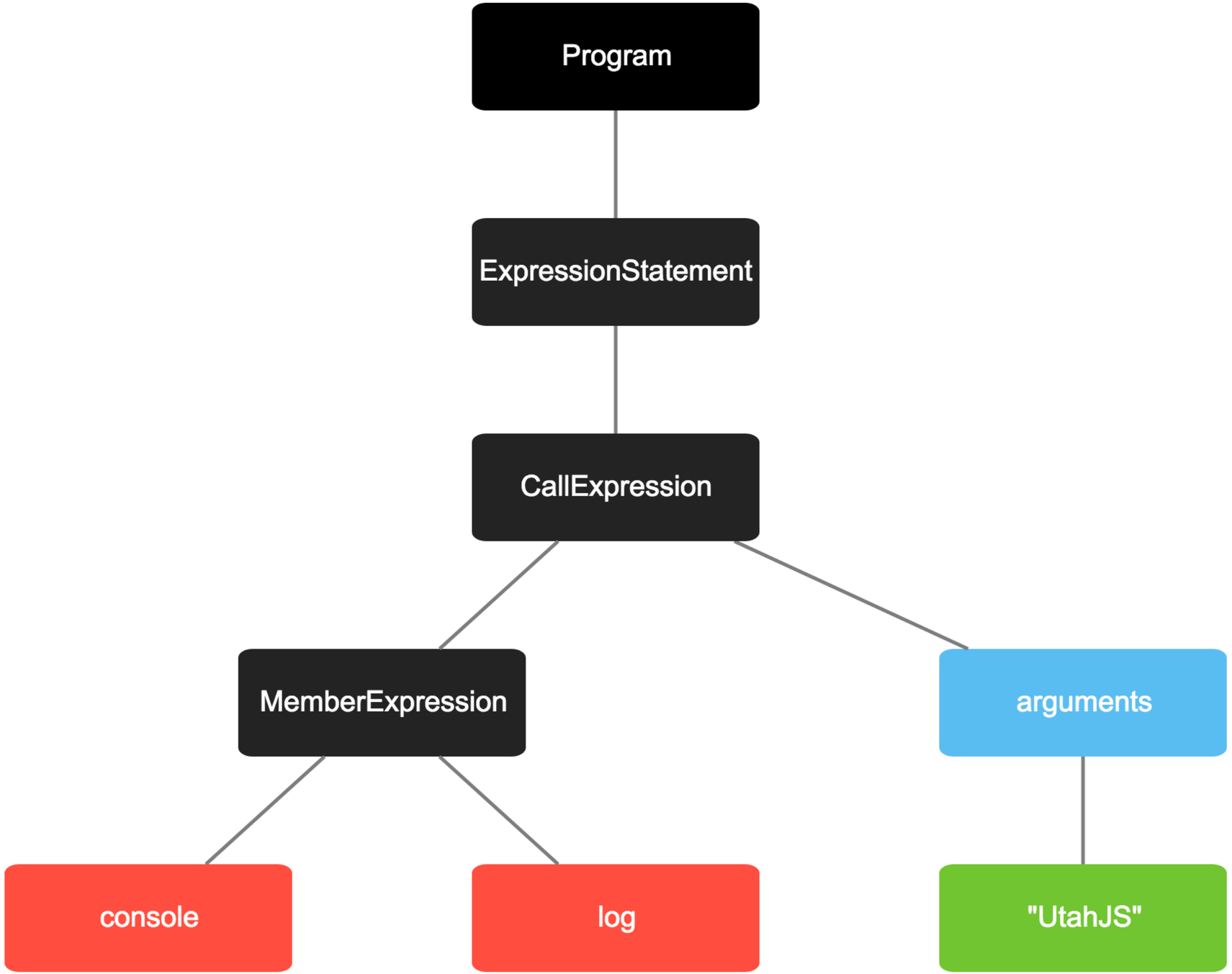

HARNESSING THE POWER OF
Abstract Syntax Trees

Parsons

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


Program
ExpressionStatement
CallExpression
MemberExpression
arguments
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\""
                    }
                ]
            }
        }
    ]
}
```


Program
ExpressionStatement
CallExpression
MemberExpression
arguments
console
log
"UtahJS"

Acorn

Espree

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)");
```


ASTS ARE
EVERYWHERE

AN *AST* 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

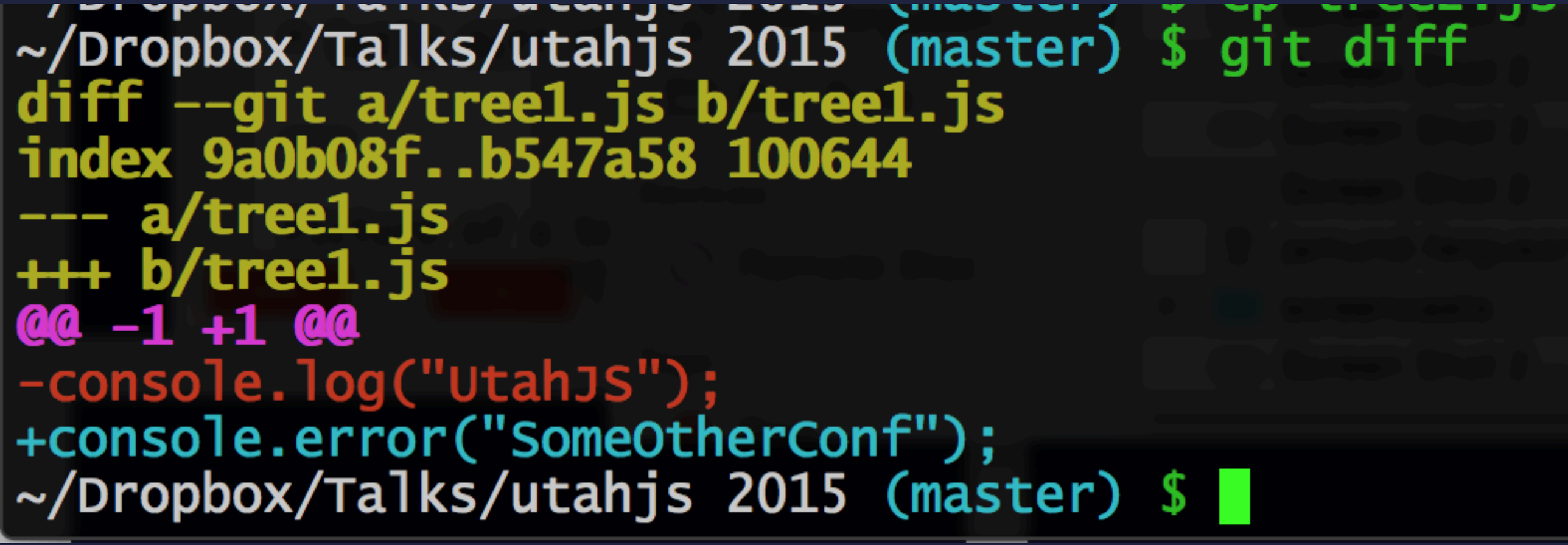
JS AWARE

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
```

```
// 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)];
});
```

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

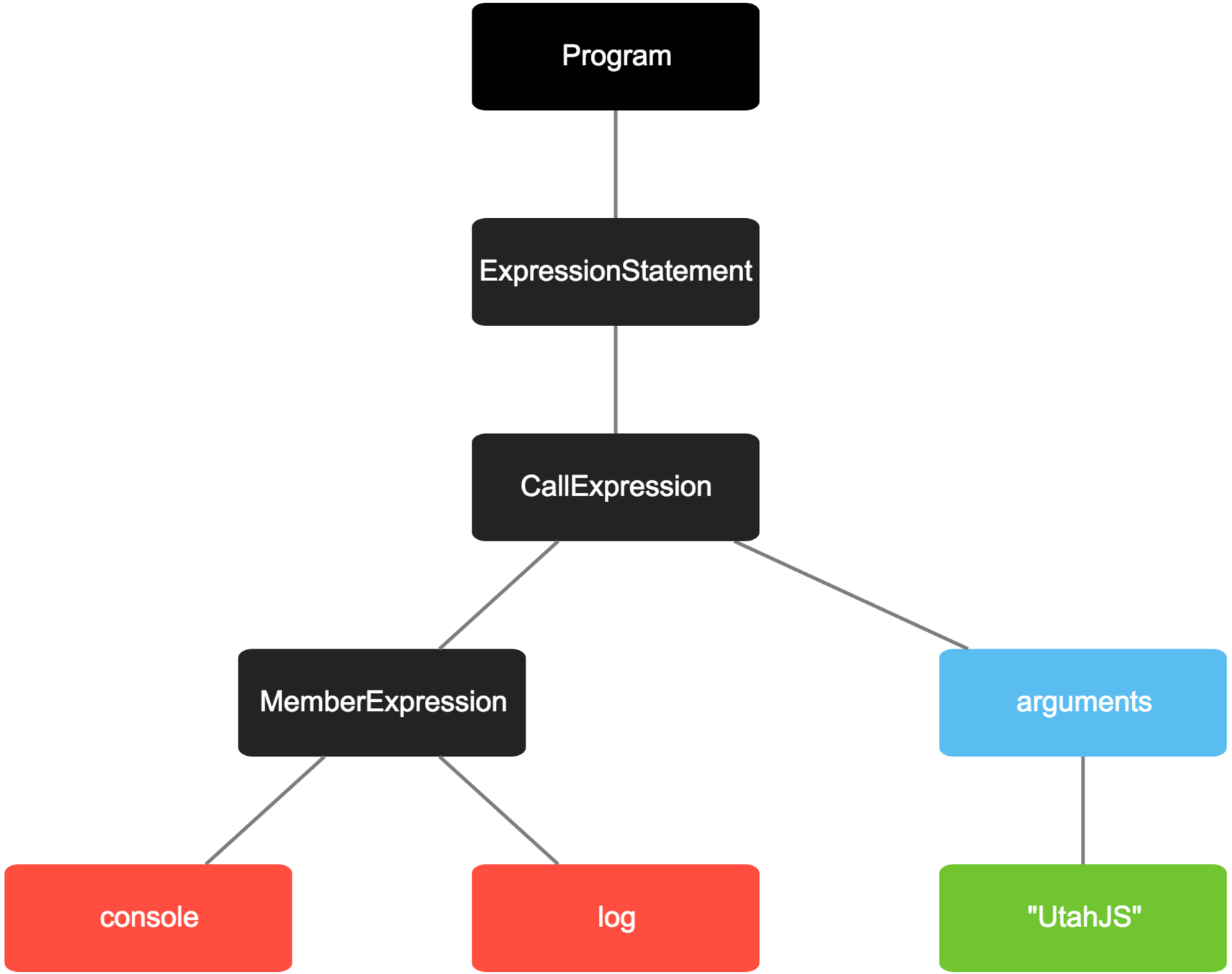
PARSING

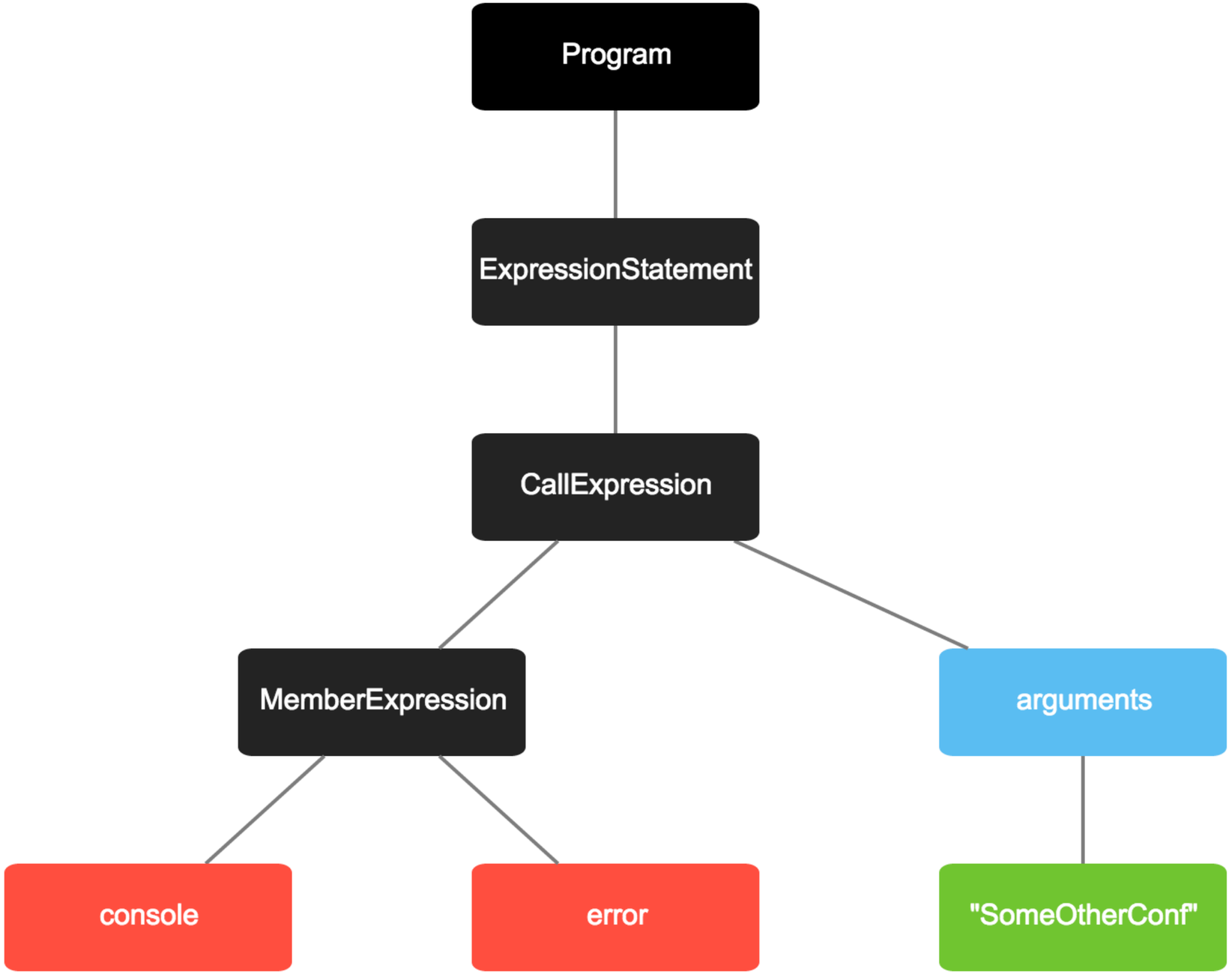
```
// generate an AST from a string of code
espree.parse("console.log('UtahJS)");
```

```
.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)
    };
})
```

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

```
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)
    };
});
```


Program
ExpressionStatement
CallExpression
MemberExpression
arguments
console
log
"UtahJS"


Program
ExpressionStatement
CallExpression
MemberExpression
arguments
console
error
"SomeOtherConf"

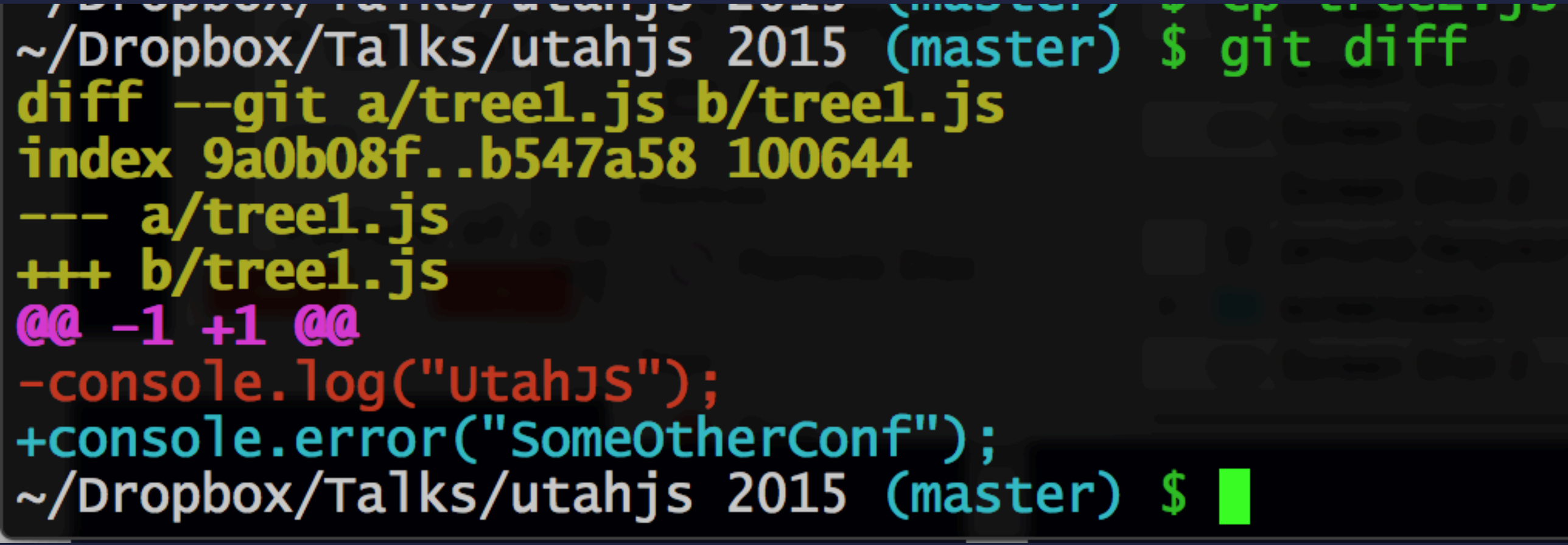
Step 2

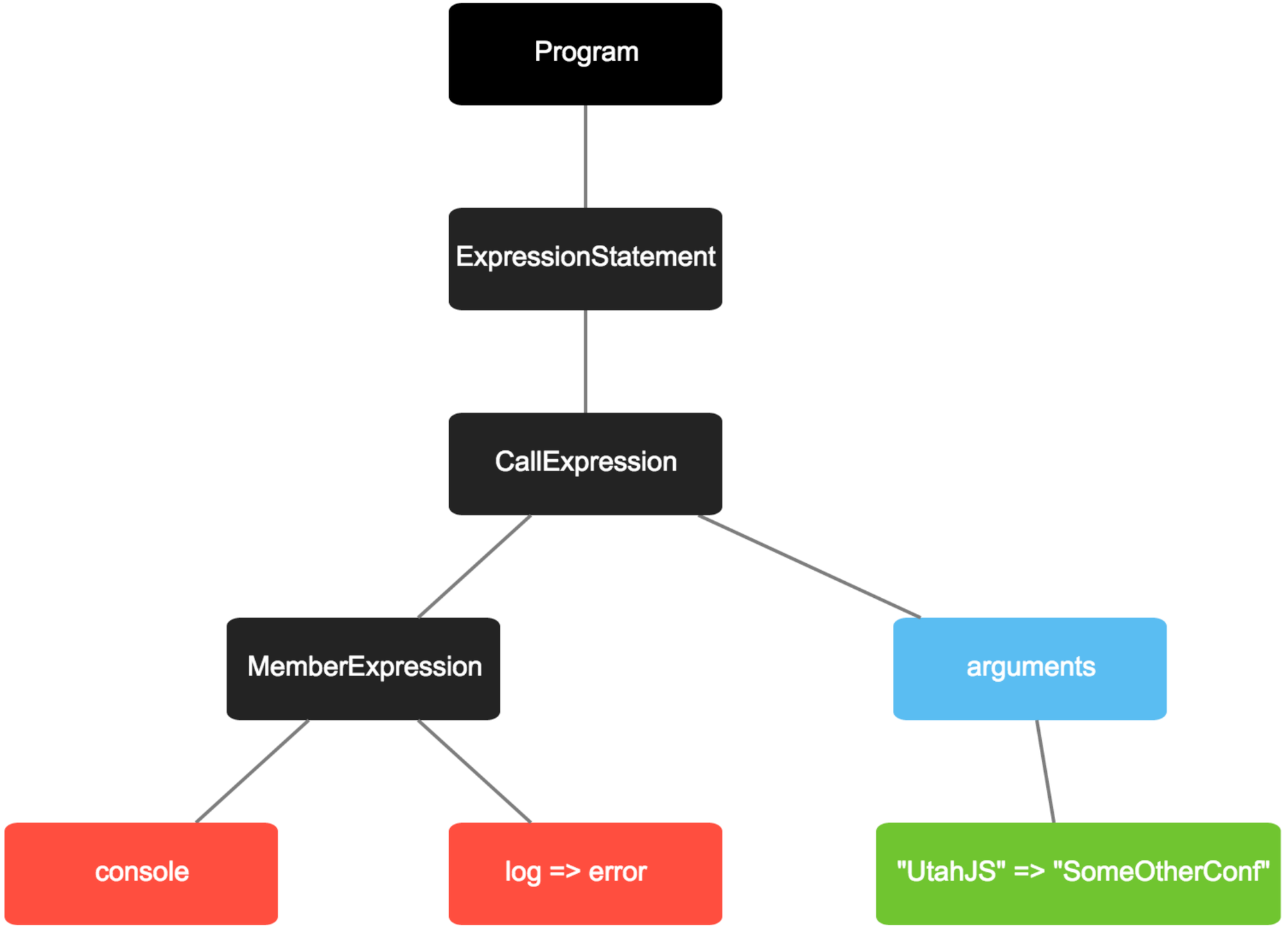
```
// let's see if something changed
var different = deepEqual(treeBefore, treeAfter);
```

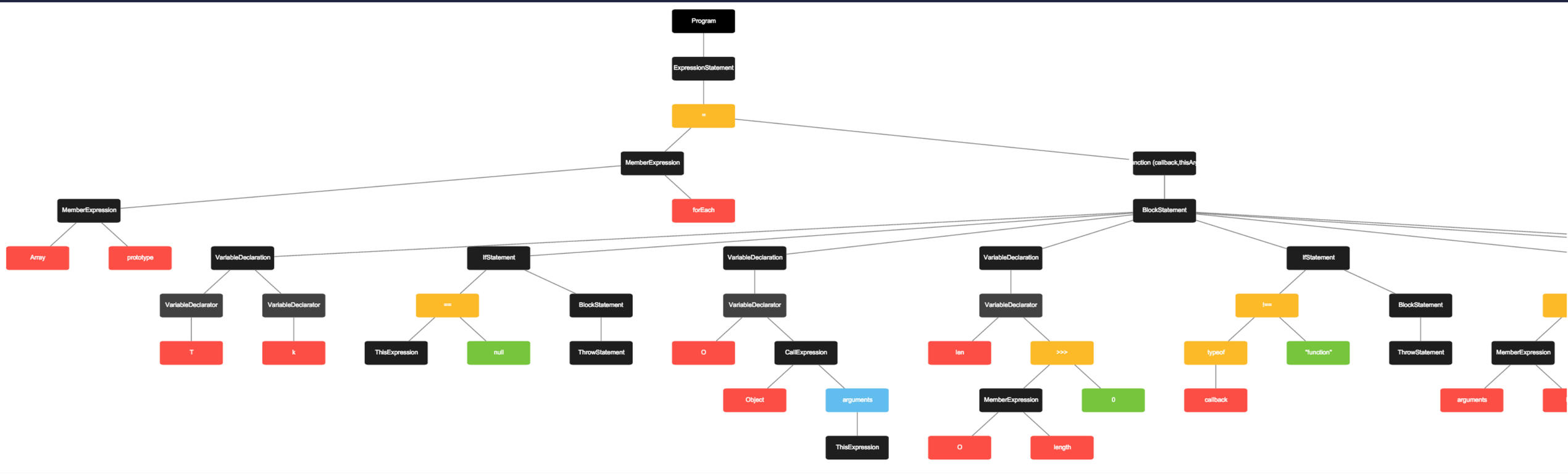
```
function deepEqual(a, b) {
    if (a === b) {
        return true;
    }
    if (!a || !b) {
        return false;
    }
    if (Array.isArray(a)) {
        return a.every(function(item, i) {
            return deepEqual(item, b[i]);
        });
    }
    if (typeof a === 'object') {
        return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

```
if (typeof a === 'object') {
    var equal = Object.keys(a).every(function(key) {
        return deepEqual(a[key], b[key]);
    });
    if (!equal) {
         // log the type of any nodes that aren't equal
        console.log('[' + a.type + '] => [' + b.type + ']');
    }
    return equal;
}

// log the any raw values that aren't equal
console.log('"' + a + '" => "' + b + '"');
```

```
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) $


Program
ExpressionStatement
CallExpression
MemberExpression
arguments
console
log => error
"UtahJS" => "SomeOtherConf"


Program
ExpressionStatement
=
MemberExpression
nction (callback,thisAr
MemberExpression
forEach
BlockStatement
Array
prototype
VariableDeclaration
IfStatement
VariableDeclaration
VariableDeclaration
IfStatement
VariableDeclarator
VariableDeclarator
==
BlockStatement
VariableDeclarator
VariableDeclarator
!==
BlockStatement
T
k
ThisExpression
null
ThrowStatement
O
CallExpression
len
>>>
typeof
"function"
ThrowStatement
MemberExpression
Object
arguments
MemberExpression
0
callback
arguments
ThisExpression
O
length

USEFUL THINGS TO DETECT

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

LET ME TELL YOU ABOUT BUILDING A HOUSE

```
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 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;
    }

});
```

INSPECTING FUNCTION DECLARATIONS

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

UNPACKING 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`;

    }, "");
}
```

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 "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 went from a return to a callback.
2. The private `getPermits` function was added.


TREES ARE SUPER
POWERFUL
INE CINEMA
me Warner Company

GETTING STARTED

2 things we need to know

Tooling

What does the JS AST look like?

How many of you use babel?

Merge pull request #3 from hzoo/patch-1 ···

 **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

```
module.exports = function (Babel) {
  return new Babel.Plugin("plugin-example", {
    visitor: {
      FunctionDeclaration: function (node, parent) {
        var id = node.id;
        node.type = "FunctionExpression";
        node.id   = null;

        return Babel.types.variableDeclaration("var", [
          Babel.types.variableDeclarator(id, node)
        ]);
      }
    }
  });
}
```

QUEST2IONS?

DETERMINING OUTPUT TYPE

```
function getOutputType(node) {
    var params = node.params.map(function(param) {
        return param.name;
    });
    var hasCallback = params[params.length - 1] === 'callback';
    var body = node.body.body || node.body; // usually child is a BlockStatement node
    var hasReturn = body.some(function(node) {
        return node.type === 'ReturnStatement';
    });
    var returnOrExecute = hasReturn ? 'return' : '';
    return hasCallback ? 'callback' : returnOrExecute;
}
```