Spying on your Code
Instrumentation in Node.js

Hi everyone!

I'm Ehden

node agent engineer, Contrast Security

github.com/cixel

ehden@contrastsecurity.com

CONTRAST SECURITY®

Hi everyone!

I'm Ehden

node agent engineer, Contrast Security

Towson alum!

github.com/cixel

ehden@contrastsecurity.com

CONTRAST SECURITY®

Instrumentation in Node.js

JavaScipt. On the server.

```
document.getElementByID('snippet')
```


node
JS®

- Assertion Testing
- Async Hooks
- Buffer
- C++ Addons
- C/C++ Addons - N-API
- Child Processes
- Cluster
- Command Line Options
- Console
- Crypto
- Debugger
- Deprecated APIs
- DNS
- Domain
- ECMAScript Modules
- Errors
- Events
- File System
- Globals
- HTTP
- HTTP/2
- HTTPS
- Inspector
- Internationalization
- Modules
- Net
- OS
- Path
- Performance Hooks
- Process
- ...

```
const thing = require('./thing');

const thingy = thing(1);

module.exports = {
    thingy
};
```

```
(function (exports, require, module, __filename, __dirname) {
    const thing = require('./thing');

    const thingy = thing(1);

    module.exports = {
        thingy
    };
});
```

```
// demo.js
const express = require('express');
const app = express();

app.get('/hi', function(req, res) {
  const name = req.query.name;
  res.send('hello, ' + encodeURIComponent(name));
});

app.listen(3000, function() {
    console.log('Example app listening on port 3000!');
});
```

```
// demo.js
const express = require('express');
const app = express();

app.get('/hi', function(req, res) {
  const name = req.query.name;
  res.send('hello, ' + encodeURIComponent(name));
});

app.listen(3000, function() {
    console.log('Example app listening on port 3000!');
});
```

```
// demo.js
const express = require('express');
const app = express();

app.get('/hi', function(req, res) {
  const name = req.query.name;
  res.send('hello, ' + encodeURIComponent(name));
});

app.listen(3000, function() {
    console.log('Example app listening on port 3000!');
});
```

```
// demo.js
const express = require('express');
const app = express();

app.get('/hi', function(req, res) {
  const name = req.query.name;
  res.send('hello, ' + encodeURIComponent(name));
});

app.listen(3000, function() {
    console.log('Example app listening on port 3000!');
});
```

```
// demo.js
const express = require('express');
const app = express();

app.get('/hi', function(req, res) {
  const name = req.query.name;
  res.send('hello, ' + encodeURIComponent(name));
});

app.listen(3000, function() {
    console.log('Example app listening on port 3000!');
});
```

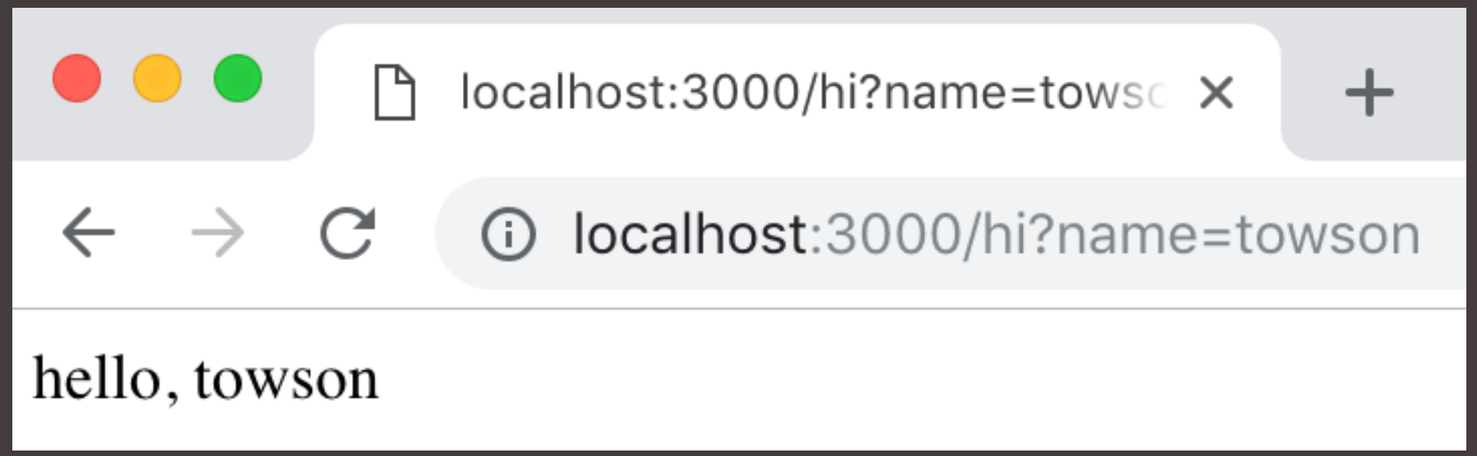
```
// demo.js
const express = require('express');
const app = express();

app.get('/hi', function(req, res) {
  const name = req.query.name;
  res.send('hello, ' + encodeURIComponent(name));
});

app.listen(3000, function() {
    console.log('Example app listening on port 3000!');
});
```

```
$ node demo.js
Example app listening on port 3000!
```

```
$ node demo.js
Example app listening on port 3000!
```



Instrumentation in ✅ Node.js

Instrumentation in ✅ Node.js

Dictionary

instrumentation

in·stru·men·ta·tion

/ˌinstrəmənˈtāSH(ə)n/

noun

noun: **instrumentation**; plural noun: **instrumentations**

1. the particular instruments used in a piece of music; the manner in which a piece is arranged for instruments.
   "Telemann's specified instrumentation of flute, violin, and continuo"
   - the arrangement or composition of a piece of music for particular musical instruments.
     "an experiment in instrumentation"
2. measuring instruments regarded collectively.
   "the controls and instrumentation of an aircraft"
   - the design, provision, or use of measuring instruments.

Translate instrumentation to Choose language

Use over time for: instrumentation

Show less

16

Feedback

📏🔬📐

Dictionary

instrumentation

in·stru·men·ta·tion

/ˌinstrəmənˈtāSH(ə)n/

noun

noun: **instrumentation**; plural noun: **instrumentations**

1. the particular instruments used in a piece of music, the manner in which a piece is arranged for instruments.
   "Telemann's specific instrumentation of flute, violin, and continuo"
   - the arrangement or composition of a piece of music for particular musical instruments.
     "an experiment in instrumentation"
2. measuring instruments regarded collectively.
   "the controls and instrumentation of an aircraft"
   - the design, provision, or use of measuring instruments.

Translate instrumentation to Choose language

Use over time for: instrumentation

Show less

17

Feedback

Dictionary

instrument

in·stru·ment

/ˈinstrəmənt/

noun

1. a tool or implement, especially one for delicate or scientific work.
   "a surgical instrument"
   synonyms: implement, tool, utensil; More
2. a measuring device used to gauge the level, position, speed, etc., of something, especially a motor vehicle or aircraft.
   synonyms: measuring device, gauge, meter; More

Instrumentation

Used for...

- logging
- performance monitoring
- debugging

Instrumentation

Used for...

- logging
- performance monitoring
- debugging

Facts about Jeff Williams:

Facts about Jeff Williams:

- Cofounder and CTO of Contrast

Facts about Jeff Williams:

- Cofounder and CTO of Contrast
- Has like 80 degrees

Facts about Jeff Williams:

- Cofounder and CTO of Contrast
- Has like 80 degrees
- Was the global OWASP chair for almost a decade

Facts about Jeff Williams:

- Cofounder and CTO of Contrast
- Has like 80 degrees
- Was the global OWASP chair for almost a decade
- Knows a ton about pen testing and application security

Facts about Jeff Williams:

- Cofounder and CTO of Contrast
- Has like 80 degrees
- Was the global OWASP chair for almost a decade
- Knows a ton about pen testing and application security
- Given a crazy amount of talks about appsec

Lasting impressions about Jeff Williams:

- *"tall enough to catch a goose with a rake"* —an actual person

Bonus fact:

- loves basketball

Bonus fact:

- loves basketball

Bonus bonus fact:

- doesn't know he's in these slides, please don't tell him

This is not just a basketball.

This is also a basketball coach.

This is not just a basketball.

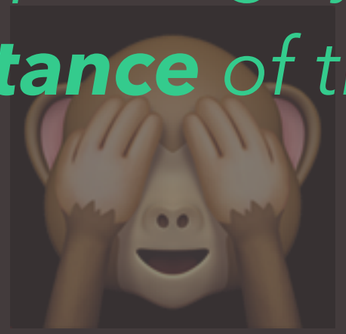
This is also a basketball coach.

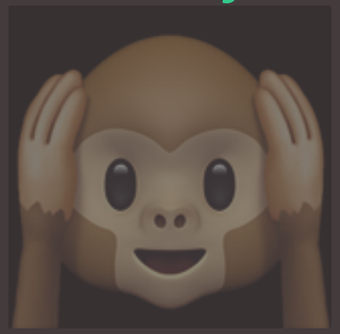
The next 20 minutes of your life

- Monkey patches
- AST rewrites
- Object proxies

Monkey patching

A monkey patch is a way for a program to ***extend or modify*** *supporting system software locally (affecting* ***only the running instance*** *of the program).*[1]



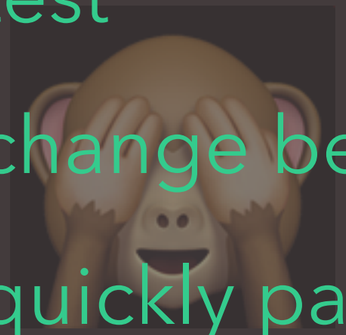


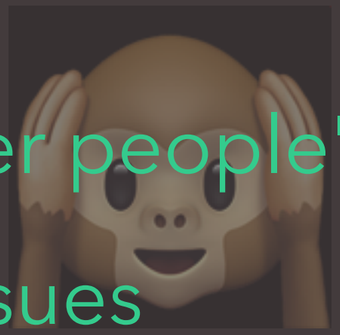
[1] Wikipedia

Monkey patching

Used to...

- test
- change behavior of other people's code
- quickly patch security issues





```
16:13:31  demo git:(master) $  node
> const oldLog = console.log
undefined
> console.log = function() {
... arguments[0] = String(arguments[0]).split('').reverse().join('');
... return oldLog.apply(this, arguments);
... }
[Function]
> console.log('hello, world')
dlrow ,olleh
undefined
```

```
// demo.js
const express = require('express');
const app = express();

app.get('/hi', function(req, res) {
  const name = req.query.name;
  res.send('hello, ' + name);
});

app.listen(3000, function() {
    console.log('Example app listening on port 3000!');
});
```

```
// demo.js
const express = require('express');
const app = express();

app.get('/hi', function(req, res) {
  const name = req.query.name;
  res.send('hello, ' + name);
});

app.listen(3000, function() {
    console.log('Example app listening on port 3000!');
});
```

```
// instrumentation/monkey.js
const Module = require('module');

const compile = Module.prototype._compile;
Module.prototype._compile = function(content, filename) {
    console.log(filename);
    const r = compile.apply(this, arguments);
    return r;
}
```

```
// instrumentation/monkey.js
const Module = require('module');

const compile = Module.prototype._compile;
Module.prototype._compile = function(content, filename) {
    console.log(filename);
    const r = compile.apply(this, arguments);
    return r;
}
```

```
// instrumentation/monkey.js
const Module = require('module');

const compile = Module.prototype._compile;
Module.prototype._compile = function(content, filename) {
    console.log(filename);
    const r = compile.apply(this, arguments);
    return r;
}
```

```
$ node -r ../instrumentation/monkey.js demo.js
```

```
/Users/ehdens/Documents/slides/node-i13n/demo/node_modules/debug/src/index.js
/Users/ehdens/Documents/slides/node-i13n/demo/node_modules/debug/src/node.js
/Users/ehdens/Documents/slides/node-i13n/demo/node_modules/debug/src/debug.js
/Users/ehdens/Documents/slides/node-i13n/demo/node_modules/ms/index.js
/Users/ehdens/Documents/slides/node-i13n/demo/node_modules/encodeurl/index.js
/Users/ehdens/Documents/slides/node-i13n/demo/node_modules/escape-html/index.js
/Users/ehdens/Documents/slides/node-i13n/demo/node_modules/on-finished/index.js
/Users/ehdens/Documents/slides/node-i13n/demo/node_modules/ee-first/index.js
/Users/ehdens/Documents/slides/node-i13n/demo/node_modules/parseurl/index.js
/Users/ehdens/Documents/slides/node-i13n/demo/node_modules/statuses/index.js
/Users/ehdens/Documents/slides/node-i13n/demo/node_modules/unpipe/index.js
/Users/ehdens/Documents/slides/node-i13n/demo/node_modules/express/lib/router/index.js
/Users/ehdens/Documents/slides/node-i13n/demo/node_modules/express/lib/router/route.js
/Users/ehdens/Documents/slides/node-i13n/demo/node_modules/array-flatten/array-flatten.js
/Users/ehdens/Documents/slides/node-i13n/demo/node_modules/express/lib/router/layer.js
/Users/ehdens/Documents/slides/node-i13n/demo/node_modules/path-to-regexp/index.js
/Users/ehdens/Documents/slides/node-i13n/demo/node_modules/methods/index.js
/Users/ehdens/Documents/slides/node-i13n/demo/node_modules/utils-merge/index.js
/Users/ehdens/Documents/slides/node-i13n/demo/node_modules/setprototypeof/index.js
/Users/ehdens/Documents/slides/node-i13n/demo/node_modules/express/lib/middleware/init.js
/Users/ehdens/Documents/slides/node-i13n/demo/node_modules/express/lib/middleware/query.js
/Users/ehdens/Documents/slides/node-i13n/demo/node_modules/qs/lib/index.js
/Users/ehdens/Documents/slides/node-i13n/demo/node_modules/qs/lib/stringify.js
/Users/ehdens/Documents/slides/node-i13n/demo/node_modules/qs/lib/utils.js
/Users/ehdens/Documents/slides/node-i13n/demo/node_modules/qs/lib/formats.js
/Users/ehdens/Documents/slides/node-i13n/demo/node_modules/qs/lib/parse.js
/Users/ehdens/Documents/slides/node-i13n/demo/node_modules/express/lib/view.js
/Users/ehdens/Documents/slides/node-i13n/demo/node_modules/express/lib/utils.js
/Users/ehdens/Documents/slides/node-i13n/demo/node_modules/safe-buffer/index.js
/Users/ehdens/Documents/slides/node-i13n/demo/node_modules/content-disposition/index.js
/Users/ehdens/Documents/slides/node-i13n/demo/node_modules/content-type/index.js
/Users/ehdens/Documents/slides/node-i13n/demo/node_modules/send/index.js
/Users/ehdens/Documents/slides/node-i13n/demo/node_modules/http-errors/index.js
/Users/ehdens/Documents/slides/node-i13n/demo/node_modules/inherits/inherits.js
/Users/ehdens/Documents/slides/node-i13n/demo/node_modules/destroy/index.js
/Users/ehdens/Documents/slides/node-i13n/demo/node_modules/etag/index.js
/Users/ehdens/Documents/slides/node-i13n/demo/node_modules/fresh/index.js
/Users/ehdens/Documents/slides/node-i13n/demo/node_modules/mime/mime.js
/Users/ehdens/Documents/slides/node-i13n/demo/node_modules/range-parser/index.js
/Users/ehdens/Documents/slides/node-i13n/demo/node_modules/proxy-addr/index.js
/Users/ehdens/Documents/slides/node-i13n/demo/node_modules/forwarded/index.js
/Users/ehdens/Documents/slides/node-i13n/demo/node_modules/ipaddr.js/lib/ipaddr.js
/Users/ehdens/Documents/slides/node-i13n/demo/node_modules/express/lib/request.js
/Users/ehdens/Documents/slides/node-i13n/demo/node_modules/accepts/index.js
/Users/ehdens/Documents/slides/node-i13n/demo/node_modules/negotiator/index.js
/Users/ehdens/Documents/slides/node-i13n/demo/node_modules/mime-types/index.js
/Users/ehdens/Documents/slides/node-i13n/demo/node_modules/mime-db/index.js
/Users/ehdens/Documents/slides/node-i13n/demo/node_modules/type-is/index.js
/Users/ehdens/Documents/slides/node-i13n/demo/node_modules/media-typer/index.js
/Users/ehdens/Documents/slides/node-i13n/demo/node_modules/express/lib/response.js
/Users/ehdens/Documents/slides/node-i13n/demo/node_modules/cookie-signature/index.js
/Users/ehdens/Documents/slides/node-i13n/demo/node_modules/cookie/index.js
/Users/ehdens/Documents/slides/node-i13n/demo/node_modules/vary/index.js
/Users/ehdens/Documents/slides/node-i13n/demo/node_modules/body-parser/lib/types/json.js
/Users/ehdens/Documents/slides/node-i13n/demo/node_modules/bytes/index.js
/Users/ehdens/Documents/slides/node-i13n/demo/node_modules/body-parser/lib/read.js
```

```
// instrumentation/monkey.js
const Module = require('module');

const compile = Module.prototype._compile;
Module.prototype._compile = function(content, filename) {
    console.log(filename);
    const r = compile.apply(this, arguments);
    return r;
}
```

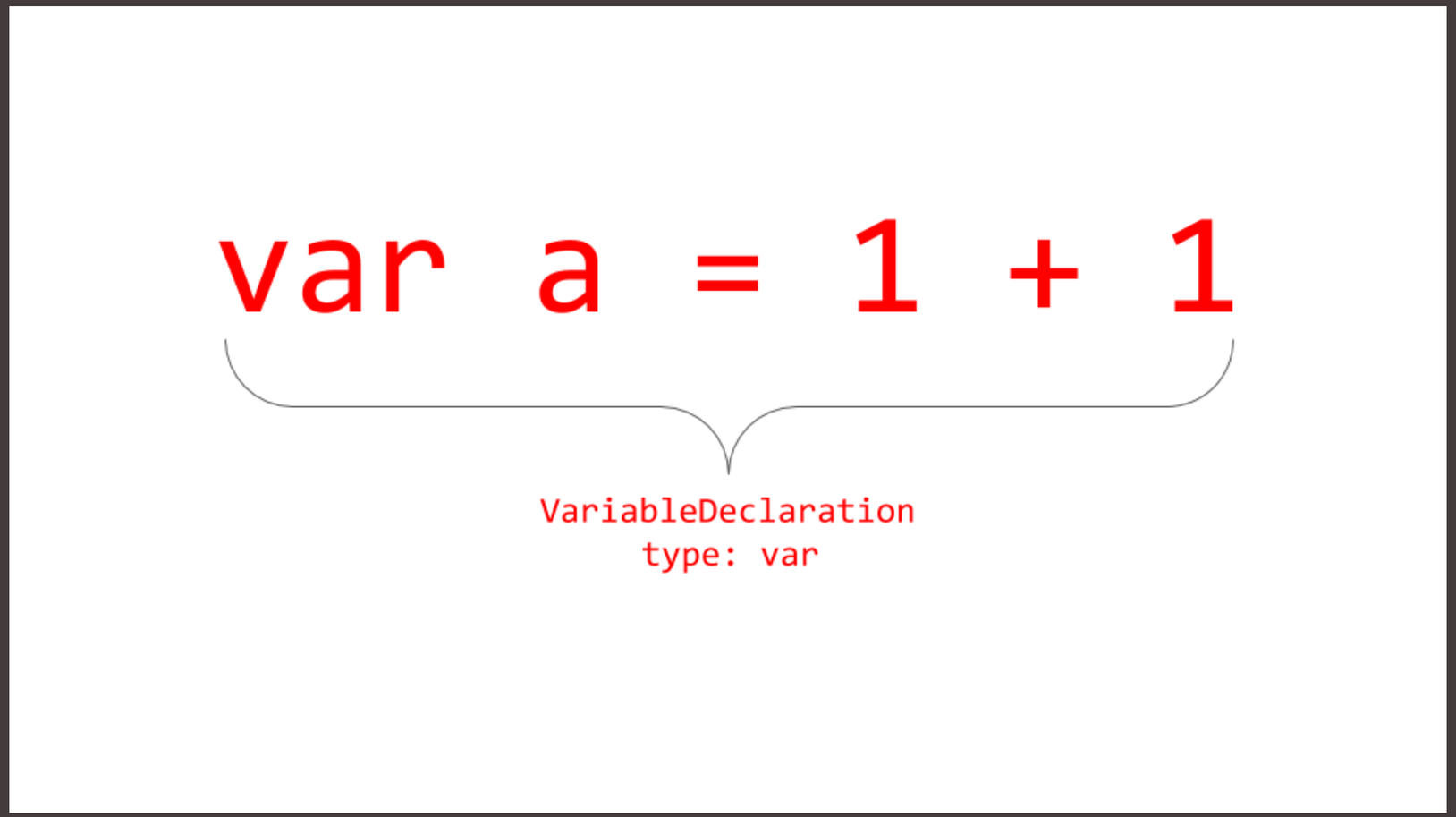
AST Rewriting

Abstract Syntax Trees

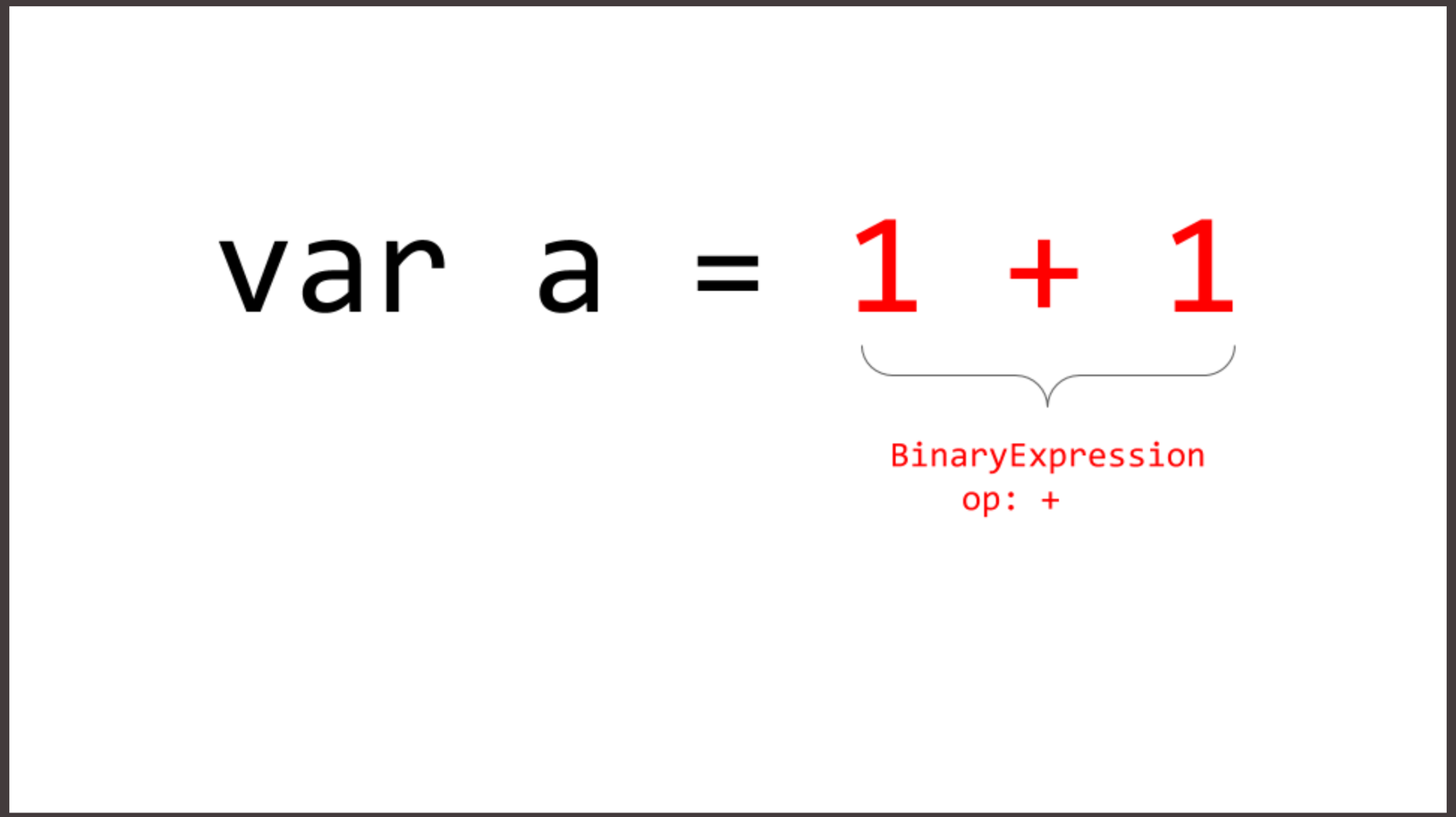
an abstract syntax tree (AST), or just syntax tree, is a tree representation of the abstract syntactic structure of source code written in a programming language.

– Wikipedia

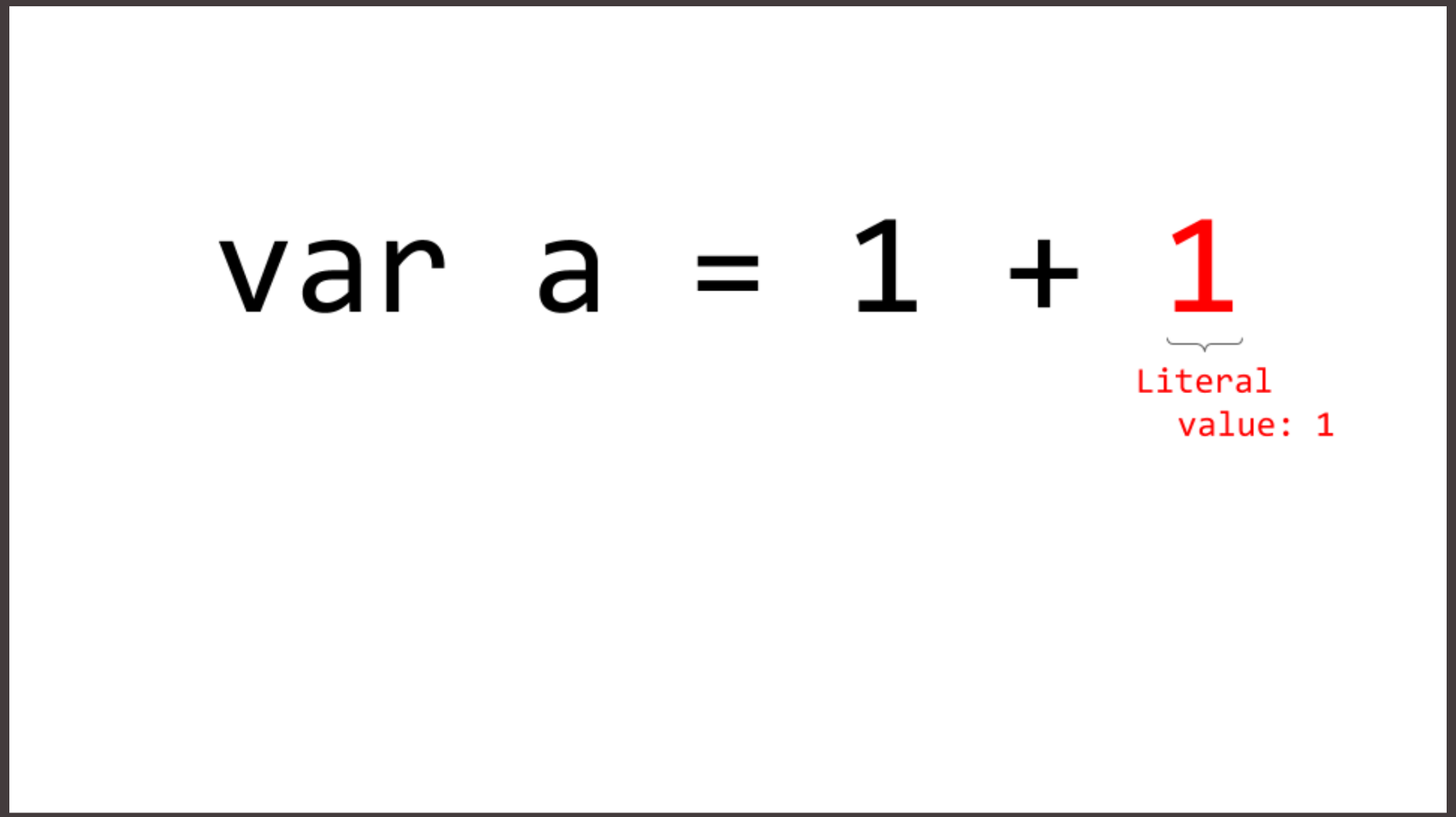
Abstract Syntax Trees



Abstract Syntax Trees



Abstract Syntax Trees



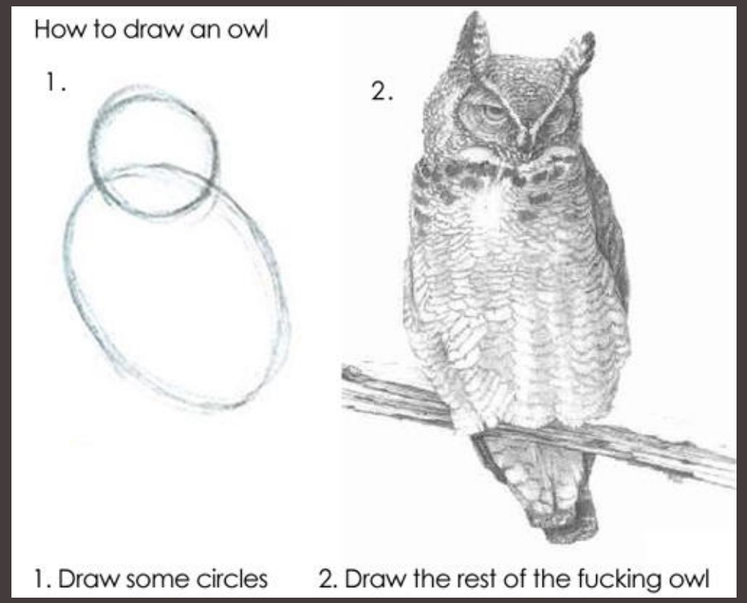
```
{
    "type": "Program",
    "body": [
        {
            "type": "VariableDeclaration",
            "declarations": [
                {
                    "type": "VariableDeclarator",
                    "id": {
                        "type": "Identifier",
                        "name": "a"
                    },
                    "init": {
                        "type": "BinaryExpression",
                        "operator": "+",
                        "left": {
                            "type": "Literal",
                            "value": 1,
                            "raw": "1"
                        },
                        "right": {
                            "type": "Literal",
                            "value": 1,
                            "raw": "1"
                        }
                    }
                }
            ],
            "kind": "var"
        }
    ],
    "sourceType": "script"
}
```


var a = 1 + 1

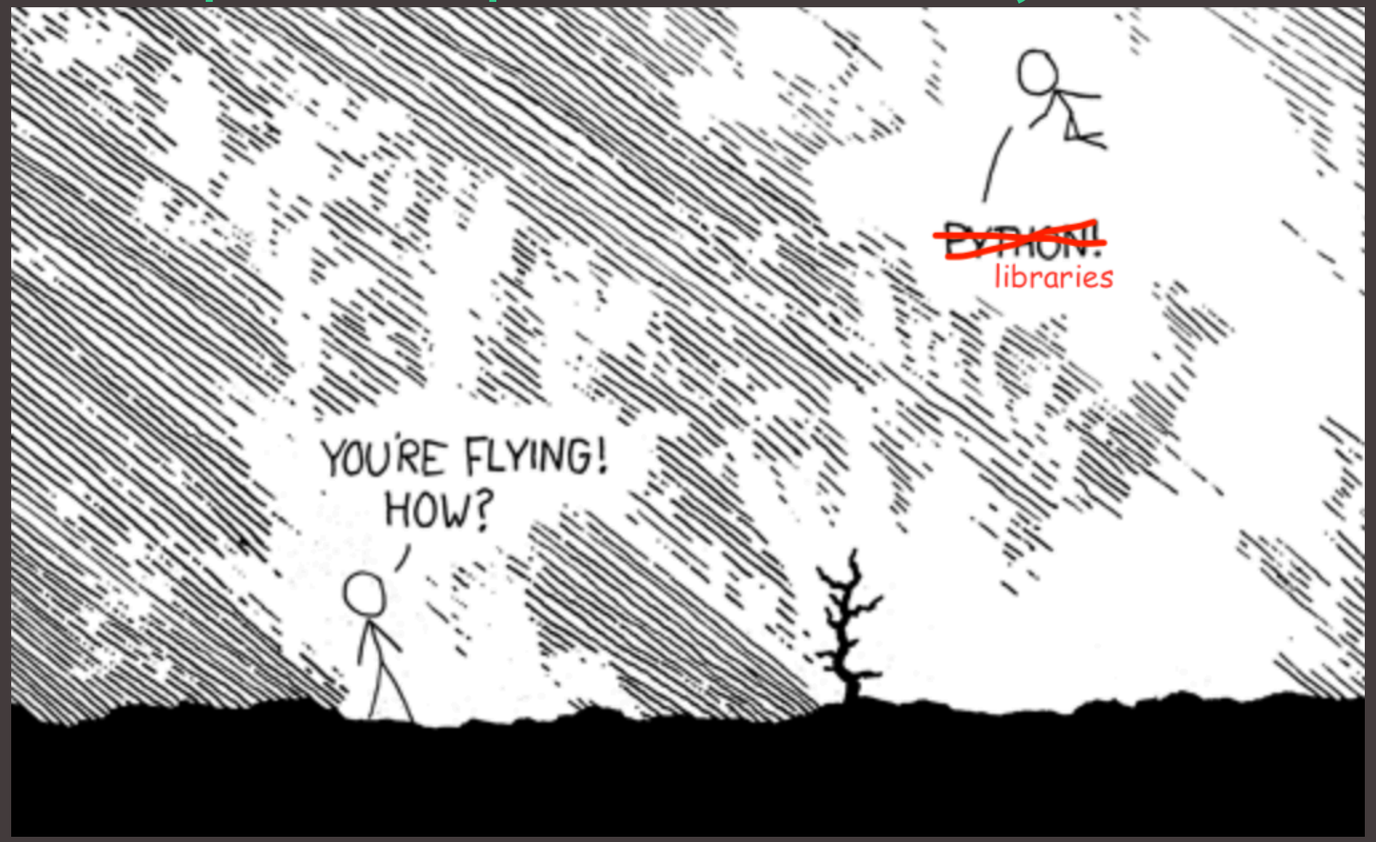

{
"type": "Program",
"body": [
{
"type": "ExpressionStatement",
"expression": {
"type": "CallExpression",
"callee": {
"type": "MemberExpression",
"computed": false,
"object": {
"type": "Identifier",
"name": "app"
},
"property": {
"type": "Identifier",
"name": "get"
}
},
"arguments": [
{
"type": "Literal",
"value": "/hi",
"raw": "'/hi'"
},
{
"type": "FunctionExpression",
"id": null,
"params": [
{
"type": "Identifier",
"name": "req"
},
{
"type": "Identifier",
"name": "res"
}
],
"body": {
"type": "BlockStatement",
"body": [
{
"type": "VariableDeclaration",
"declarations": [
{
"type": "VariableDeclarator",
"id": {
"type": "Identifier",
"name": "name"
},
"init": {
"type": "MemberExpression",
"computed": false,
"object": {
"type": "MemberExpression",
"computed": false,
"object": {
"type": "Identifier",
"name": "req"
},
"property": {
"type": "Identifier",
"name": "query"
}
},
"property": {
"type": "Identifier",
"name": "name"
}
}
}
],
"kind": "const"
},
{
"type": "ExpressionStatement",
"expression": {
"type": "CallExpression",
"callee": {
"type": "MemberExpression",
"computed": false,
"object": {
"type": "Identifier",
"name": "res"
},
"property": {
"type": "Identifier",
"name": "send"
}
},
"arguments": [
{
"type": "BinaryExpression",
"operator": "+",
"left": {
"type": "Literal",
"value": "hello, ",
"raw": "'hello, '"
},
"right": {
"type": "CallExpression",
"callee": {
"type": "Identifier",
"name": "encodeURIComponent"
},
"arguments": [
{
"type": "Identifier",
"name": "name"
}
]
}
}
]
}
}
]
},
"generator": false,
"expression": false,
"async": false
}
]
}
}
],
"sourceType": "script"
}

AST Rewriting

1. parse source code into AST
2. traverse tree, changing whatever you want
3. print the tree back out as source



```
// Step 1: Parse code into AST
const ast = esprima.parse(code);
```



```
// Step 2: Traverse the AST and change what we want
const after = estraverse.replace(before, {
    enter(node) {
        switch(node.type) {
            case 'FunctionDeclaration':
                const name = node.id && node.id.name || 'anonymous';
                const start = node.loc.start;
                const rel = path.relative(process.cwd(), filename);
                const loc = `${name} (${rel}:${start.line}:${start.column})`;

                node.body.type === 'BlockStatement' && node.body.body.unshift({
                    'type': 'ExpressionStatement',
                    'expression': {
                        'type': 'CallExpression',
                        'callee': {
                            'type': 'MemberExpression',
                            'computed': false,
                            'object': {
                                'type': 'Identifier',
                                'name': 'console'
                            },
                            'property': {
                                'type': 'Identifier',
                                'name': 'log'
                            }
                        },
                        'arguments': [
                            {
                                type: 'Literal',
                                value: loc
                            }
                        ]
                    }
                });
            // ...
        }

        return node;
    }
});
```

```
// Step 3: Print the new AST
return escodegen.generate(after);
```

Monkey patch + AST rewrite

```
// instrumentation/index.js
Module.prototype._compile = function(content, filename) {
    arguments[0] = rewrite(content, filename);
    const r = compile.apply(this, arguments);
    return r;
}
```

```
pathtoRegexp (node_modules/path-to-regexp/index.js:28:0)
listen (node_modules/express/lib/application.js:616:13)
anonymous (demo.js:10:17)
Example app listening on port 3000!
anonymous (node_modules/express/lib/express.js:38:12)
handle (node_modules/express/lib/application.js:158:13)
anonymous (node_modules/express/lib/application.js:473:16)
set (node_modules/express/lib/application.js:352:10)
finalhandler (node_modules/finalhandler/index.js:77:0)
handle (node_modules/express/lib/router/index.js:136:15)
debug (node_modules/debug/src/debug.js:65:2)
getProtohost (node_modules/express/lib/router/index.js:535:0)
restore (node_modules/express/lib/router/index.js:620:0)
next (node_modules/express/lib/router/index.js:176:2)
getPathname (node_modules/express/lib/router/index.js:526:0)
parseurl (node_modules/parseurl/index.js:35:0)
```

AST Rewriting

Used for...

- React, TypeScript
- Minification
- Prettification
- Refactoring
- Code coverage

```
// demo.js
const express = require('express');
const app = express();

app.get('/hi', function(req, res) {
  const name = req.query.name;
  res.send('hello, ' + name);
});

app.listen(3000, function() {
    console.log('Example app listening on port 3000!');
});
```

```
// demo.js
const express = require('express');
const app = express();

app.get('/hi', function(req, res) {
  const name = req.query.name;
  res.send('hello, ' + name);
});

app.listen(3000, function() {
    console.log('Example app listening on port 3000!');
});
```

Proxy & Reflect

Proxy

Used to define custom behavior for fundamental operations (e.g. property lookup, assignment, enumeration, function invocation, etc).[2]

[2] MDN

Terminology

target: the object being wrapped

trap: method providing intercept behavior for an operation

handler: object which holds traps

```
const person = new Person('Jean-Luc Picard');

const handler = {
    get(target, property, receiver) {
        if (property === 'name') {
            return 'Locutus of Borg';
        }

        return Reflect.get(...arguments);
    },

    set(target, property, value, receiver) {
        // Don't actually set.
        console.log('Resistance is futile.');
        return true;
    }
};

const borg = new Proxy(person, handler);
```

```
const person = new Person('Jean-Luc Picard');

const handler = {
    get(target, property, receiver) {
        if (property === 'name') {
            return 'Locutus of Borg';
        }

        return Reflect.get(...arguments);
    },

    set(target, property, value, receiver) {
        // Don't actually set.
        console.log('Resistance is futile.');
        return true;
    }
};

const borg = new Proxy(person, handler);
```

```
const person = new Person('Jean-Luc Picard');

const handler = {
    get(target, property, receiver) {
        if (property === 'name') {
            return 'Locutus of Borg';
        }

        return Reflect.get(...arguments);
    },

    set(target, property, value, receiver) {
        // Don't actually set.
        console.log('Resistance is futile.');
        return true;
    }
};

const borg = new Proxy(person, handler);
```

```
const person = new Person('Jean-Luc Picard');

const handler = {
    get(target, property, receiver) {
        if (property === 'name') {
            return 'Locutus of Borg';
        }

        return Reflect.get(...arguments);
    },

    set(target, property, value, receiver) {
        // Don't actually set.
        console.log('Resistance is futile.');
        return true;
    }
};

const borg = new Proxy(person, handler);
```

```
const person = new Person('Jean-Luc Picard');

const handler = {
    get(target, property, receiver) {
        if (property === 'name') {
            return 'Locutus of Borg';
        }

        return Reflect.get(...arguments);
    },

    set(target, property, value, receiver) {
        // Don't actually set.
        console.log('Resistance is futile.');
        return true;
    }
};

const borg = new Proxy(person, handler);
```

```
const person = new Person('Jean-Luc Picard');

const handler = {
    get(target, property, receiver) {
        if (property === 'name') {
            return 'Locutus of Borg';
        }

        return Reflect.get(...arguments);
    },

    set(target, property, value, receiver) {
        // Don't actually set.
        console.log('Resistance is futile.');
        return true;
    }
};

const borg = new Proxy(person, handler);
```

Watching request reads

```
const name = req.query.name;
```

1. Monkey-patch express, add intercept for every incoming request
2. Proxy `request.query` with a `get` trap

Patching express

```
const express = require('express');
// patch express()?
const app = express();
```

Patching express

```
const load = Module._load;
Module._load = function(filename) {
    let mod = load.apply(this, arguments);

    if (filename === 'express') {
        // woo!
    }

    return mod;
};
```

Patching express

```
const express = require('express');
const app = express();
console.log(Object.keys(express));
/*
[
    application,
    request,
    response,
    Route,
    Router,
    json,
    query,
    static,
    urlencoded
]
*/
```

Patching express

```
let mod = load.apply(this, arguments);
if (filename === 'express') {
    mod = new Proxy(mod, {
        apply(tar, thisArg, args) {
            return tar.apply(thisArg, args);
        }
    });
}
return mod;
```

Watching requests

```
mod = new Proxy(mod, {
    apply(tar, thisArg, args) {
        const app = tar.apply(thisArg, args);

        app.use(function(req, res, next) {
            req.query = new Proxy(req.query, {
                get(tar, prop, recv) {
                    const val = Reflect.get(...arguments);
                    console.log(`getting ${prop} from query: ${val}`);
                    return val;
                }
            })
        });

        return app;
    }
});
```

Other fun stuff

```
apply(tar, thisArg, args) {
    const app = tar.apply(thisArg, args);

    app.use(function(req, res, next) {
        req.query = new Proxy(req.query, {
            get(tar, prop, recv) {
                const val = Reflect.get(...arguments);
                data.queries.push({
                    key: prop,
                    val: v
                });
                return val;
            }
        })
    });

    app.get('/intrumentation/calls', function (req, res) {
        res.send(data.calls);
    });

    return app;
}
```

Demo

Proxy stuff I should mention

Interesting patterns:

[3] Transparent Object Proxies for JavaScript

Proxy stuff I should mention

Interesting patterns:

- Recursive proxies[3]

[3] Transparent Object Proxies for JavaScript

Proxy stuff I should mention

Interesting patterns:

- Recursive proxies[3]
- Proxies on empty object

[3] Transparent Object Proxies for JavaScript

Proxy stuff I should mention

Interesting patterns:

- Recursive proxies[3]
- Proxies on empty object
- Revocable proxies

[3] Transparent Object Proxies for JavaScript

It's over

- instrumentation is awesome
- javascript is ___
- composition of monkey patches, AST rewrites, and proxies = $$

Contact me:

- ehden@contrastsecurity.com
- github.com/cixel

Image Credits

- Contrast Security: https://www.contrastsecurity.com
- Node.js Foundation: https://nodejs.org/en/about/resources/
- Slate