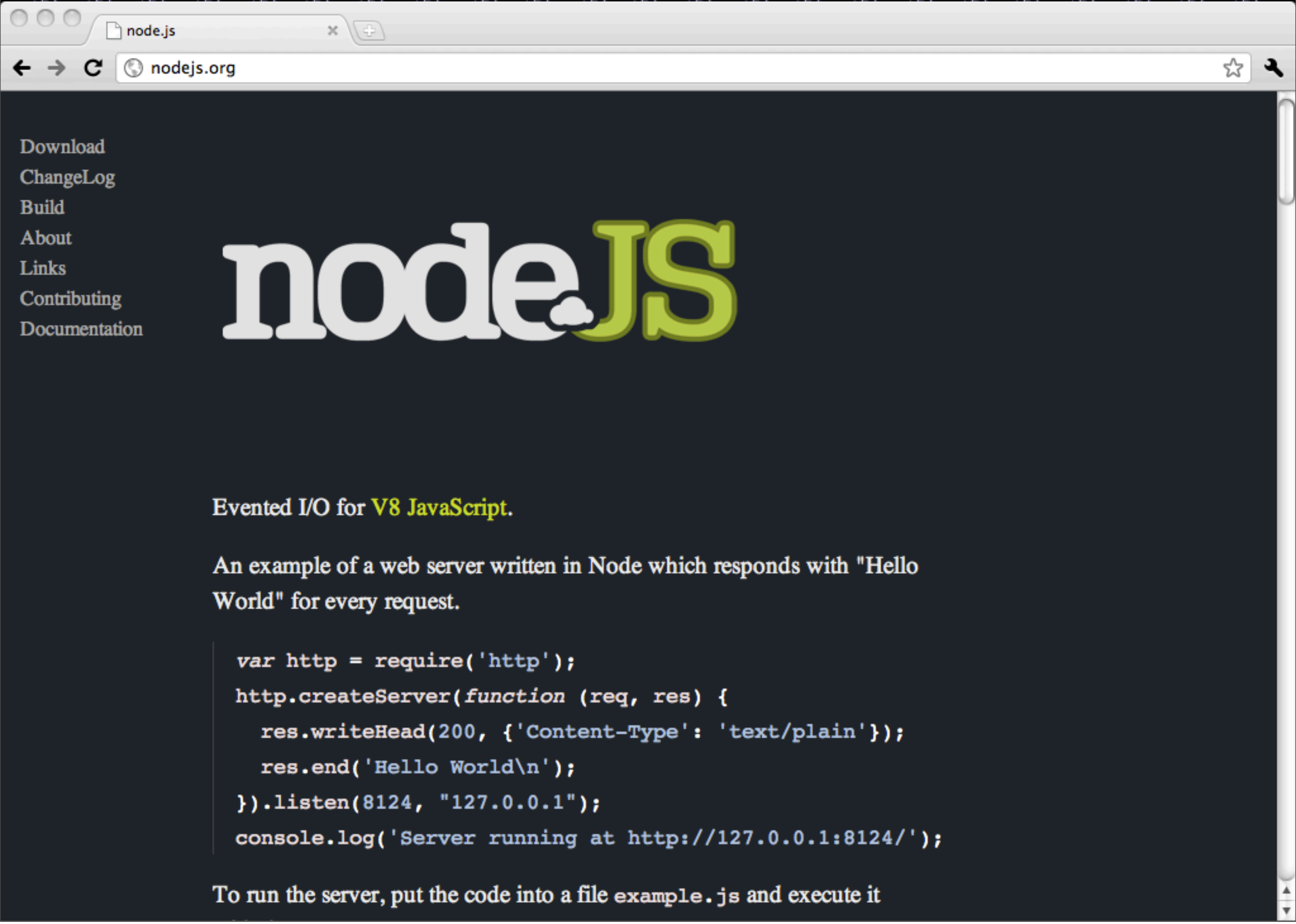
An Introduction to nodeJS

By Tim Caswell

What is Node.JS?

- Google's Super fast V8 JavaScript engine
- Highly Optimized libev and libeio C libraries
- JavaScript for the Server



Evented I/O for V8 JavaScript.

An example of a web server written in Node which responds with "Hello World" for every request.

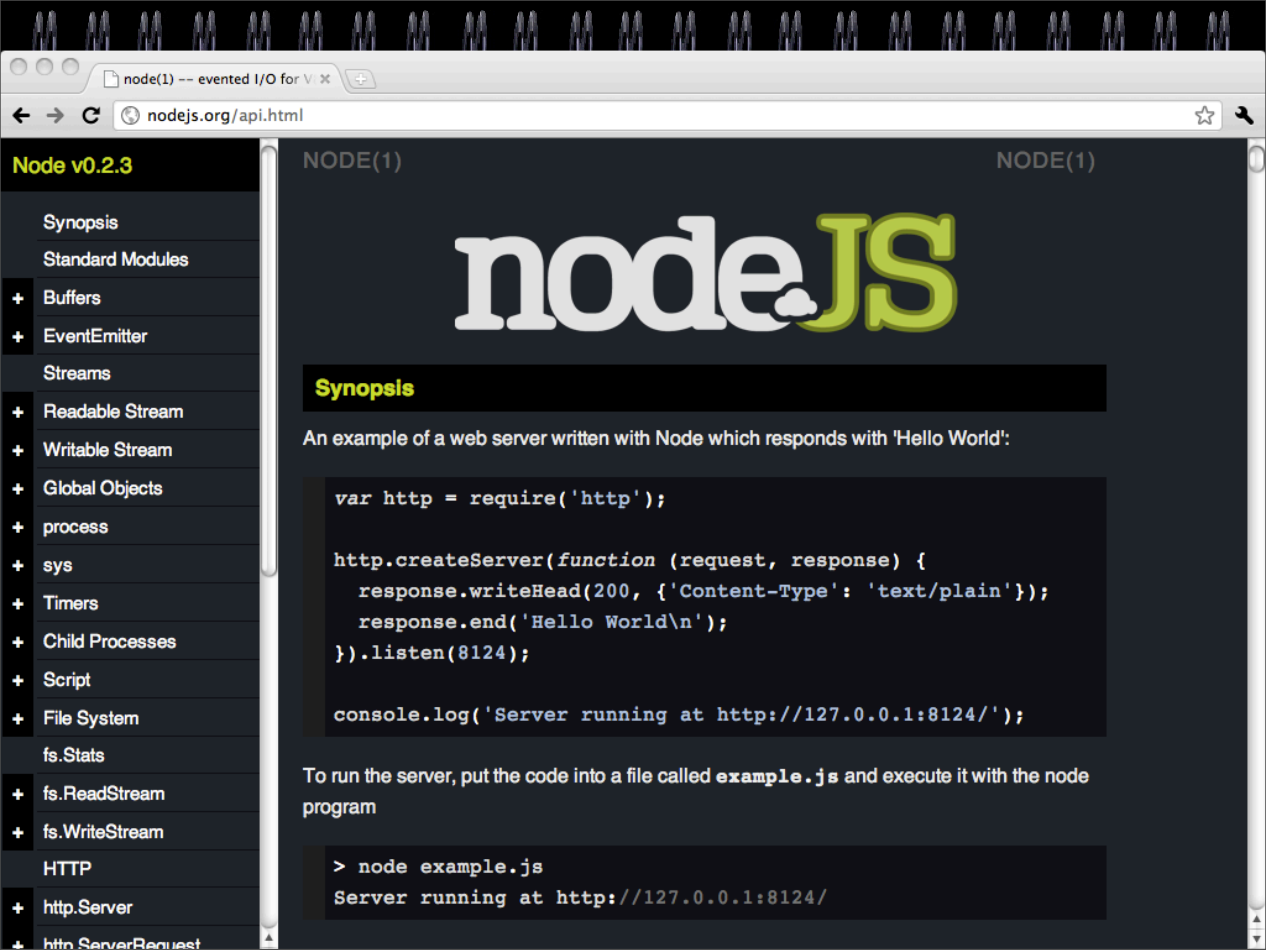
```
var http = require('http');
http.createServer(function (req, res) {
  res.writeHead(200, {'Content-Type': 'text/plain'});
  res.end('Hello World\n');
}).listen(8124, "127.0.0.1");
console.log('Server running at http://127.0.0.1:8124/');
```

To run the server, put the code into a file `example.js` and execute it

Saturday, October 9, 2010

Why is Node.JS awesome?

- It's fast, really fast!
- It's JavaScript!
- You already know it!
- It can do anything!



Node v0.2.3

NODE(1) NODE(1)

Synopsis

An example of a web server written with Node which responds with 'Hello World':

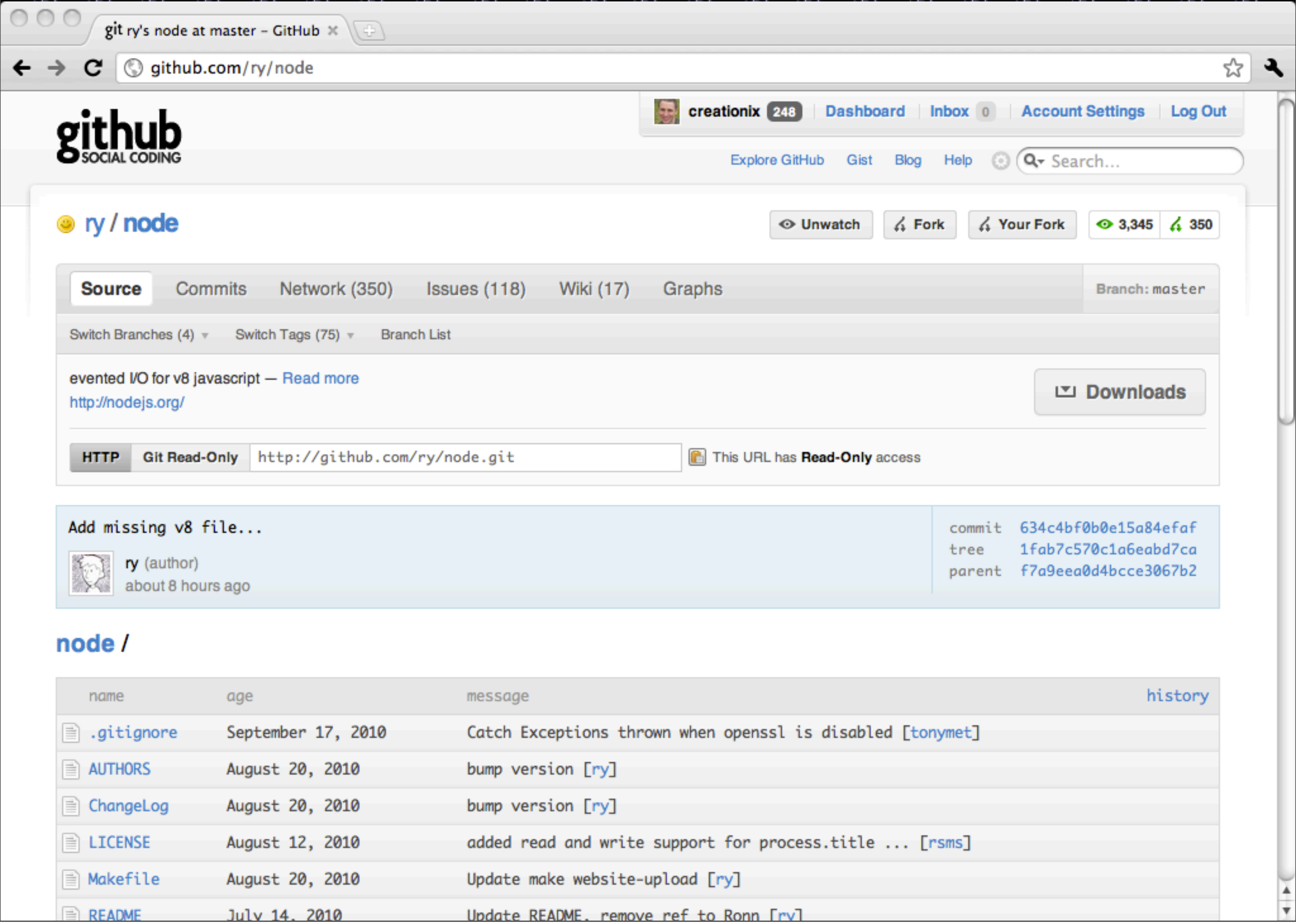
```
var http = require('http');

http.createServer(function (request, response) {
  response.writeHead(200, {'Content-Type': 'text/plain'});
  response.end('Hello World\n');
}).listen(8124);

console.log('Server running at http://127.0.0.1:8124/');
```

To run the server, put the code into a file called **`example.js`** and execute it with the node program

```
> node example.js
Server running at http://127.0.0.1:8124/
```

Why do we need awesome?

- The world is moving real-time
- Real-time require persistent connections
- Threads really suck for this.
- Event Loops are the awesome to solve this.
- JavaScript is agile.



ry / node

evented I/O for v8 javascript — Read more
http://nodejs.org/

http://github.com/ry/node.git This URL has **Read-Only** access

Add missing v8 file...
ry (author)
about 8 hours ago

```
commit  634c4bf0b0e15a84efaf
tree    1fab7c570c1a6eabd7ca
parent  f7a9eea0d4bcce3067b2
```

node /

name	age	message
.gitignore	September 17, 2010	Catch Exceptions thrown when openssl is disabled [tonymet]
AUTHORS	August 20, 2010	bump version [ry]
ChangeLog	August 20, 2010	bump version [ry]
LICENSE	August 12, 2010	added read and write support for process.title ... [rsms]
Makefile	August 20, 2010	Update make website-upload [ry]
README	July 14, 2010	Update README, remove ref to Ronn [ry]

history

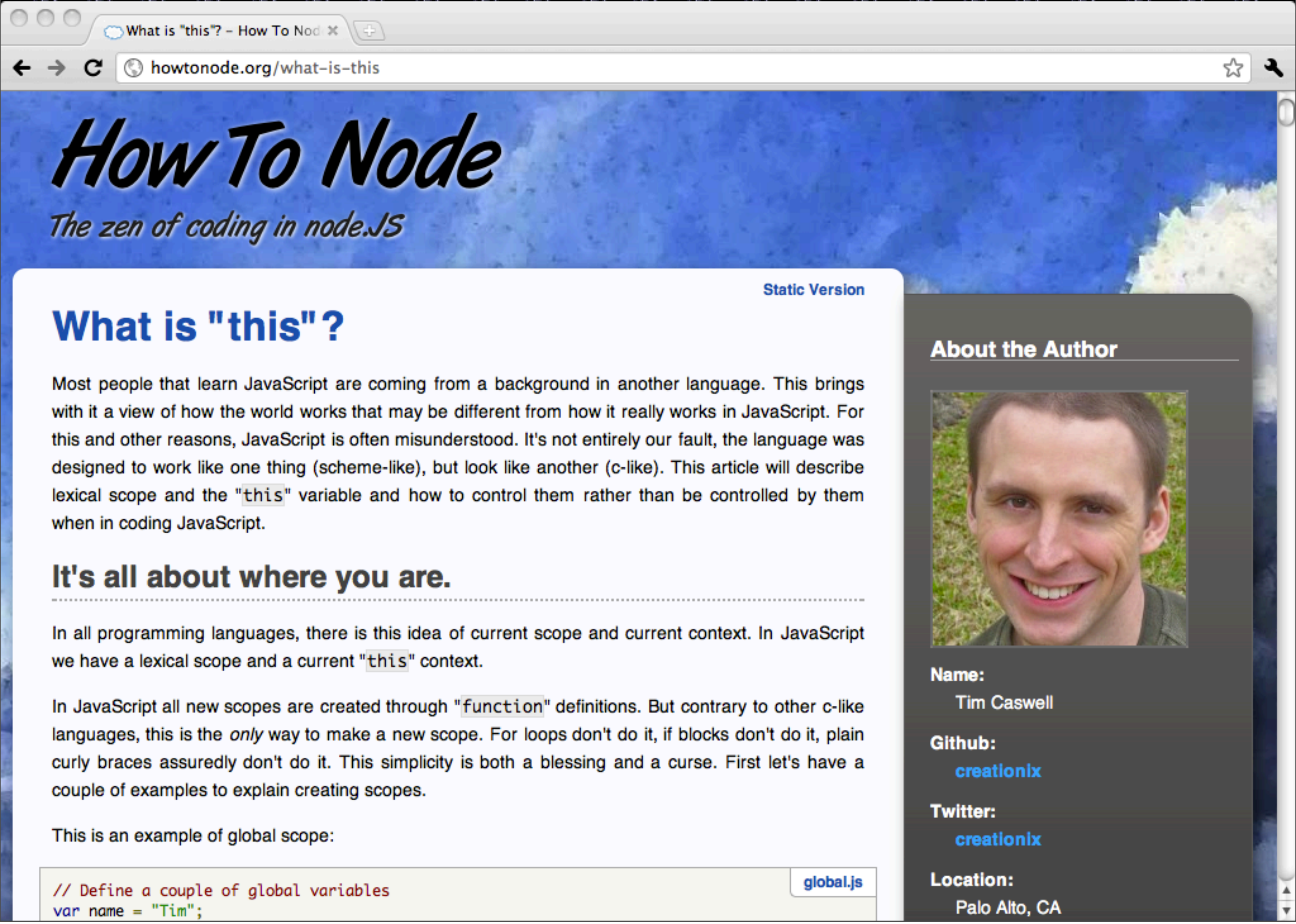
What is not Node’s model?

- ☐ Multi-threaded
- ☐ Multiple stacks
- ☐ Slow
- ☐ The same old stuff

What is the Node Model?

- Single Threaded
- Single Stack
- Non-Blocking I/O
- Event and Callback Based

Node.JS Examples



How To Node

The zen of coding in node.JS

Static Version

What is "this"?

Most people that learn JavaScript are coming from a background in another language. This brings with it a view of how the world works that may be different from how it really works in JavaScript. For this and other reasons, JavaScript is often misunderstood. It's not entirely our fault, the language was designed to work like one thing (scheme-like), but look like another (c-like). This article will describe lexical scope and the "`this`" variable and how to control them rather than be controlled by them when in coding JavaScript.

It's all about where you are.

In all programming languages, there is this idea of current scope and current context. In JavaScript we have a lexical scope and a current "`this`" context.

In JavaScript all new scopes are created through "`function`" definitions. But contrary to other c-like languages, this is the *only* way to make a new scope. For loops don't do it, if blocks don't do it, plain curly braces assuredly don't do it. This simplicity is both a blessing and a curse. First let's have a couple of examples to explain creating scopes.

This is an example of global scope:

global.js

```
// Define a couple of global variables
var name = "Tim";
```

About the Author

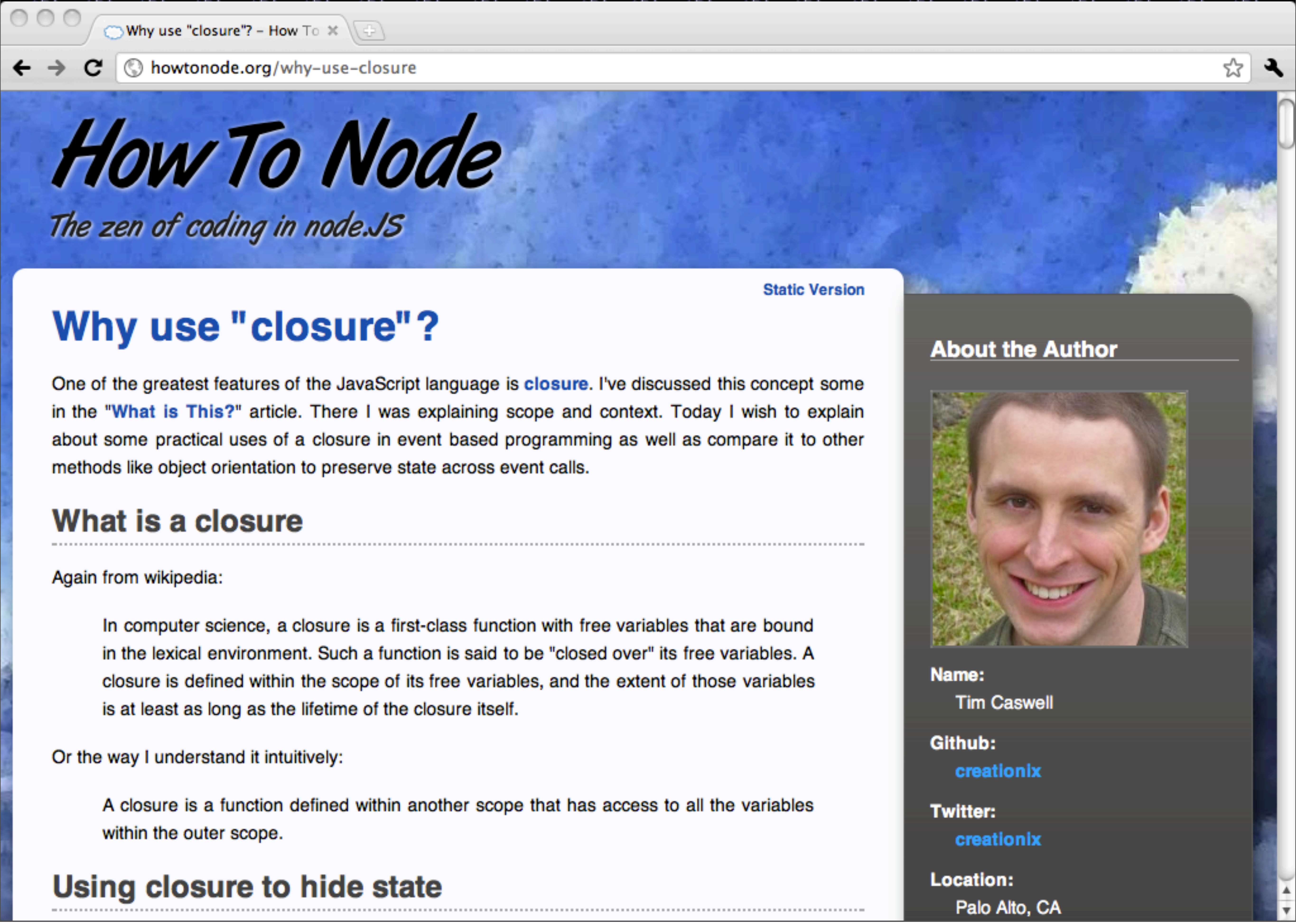
Name:
Tim Caswell

Github:
creationix

Twitter:
creationix

Location:
Palo Alto, CA

Saturday, October 9, 2010


Why use "closure"? – How To

howtonode.org/why-use-closure

How To Node

The zen of coding in node.JS

Static Version

Why use "closure"?

One of the greatest features of the JavaScript language is **closure**. I've discussed this concept some in the "**What is This?**" article. There I was explaining scope and context. Today I wish to explain about some practical uses of a closure in event based programming as well as compare it to other methods like object orientation to preserve state across event calls.

What is a closure

Again from wikipedia:

> In computer science, a closure is a first-class function with free variables that are bound in the lexical environment. Such a function is said to be "closed over" its free variables. A closure is defined within the scope of its free variables, and the extent of those variables is at least as long as the lifetime of the closure itself.

Or the way I understand it intuitively:

> A closure is a function defined within another scope that has access to all the variables within the outer scope.

Using closure to hide state

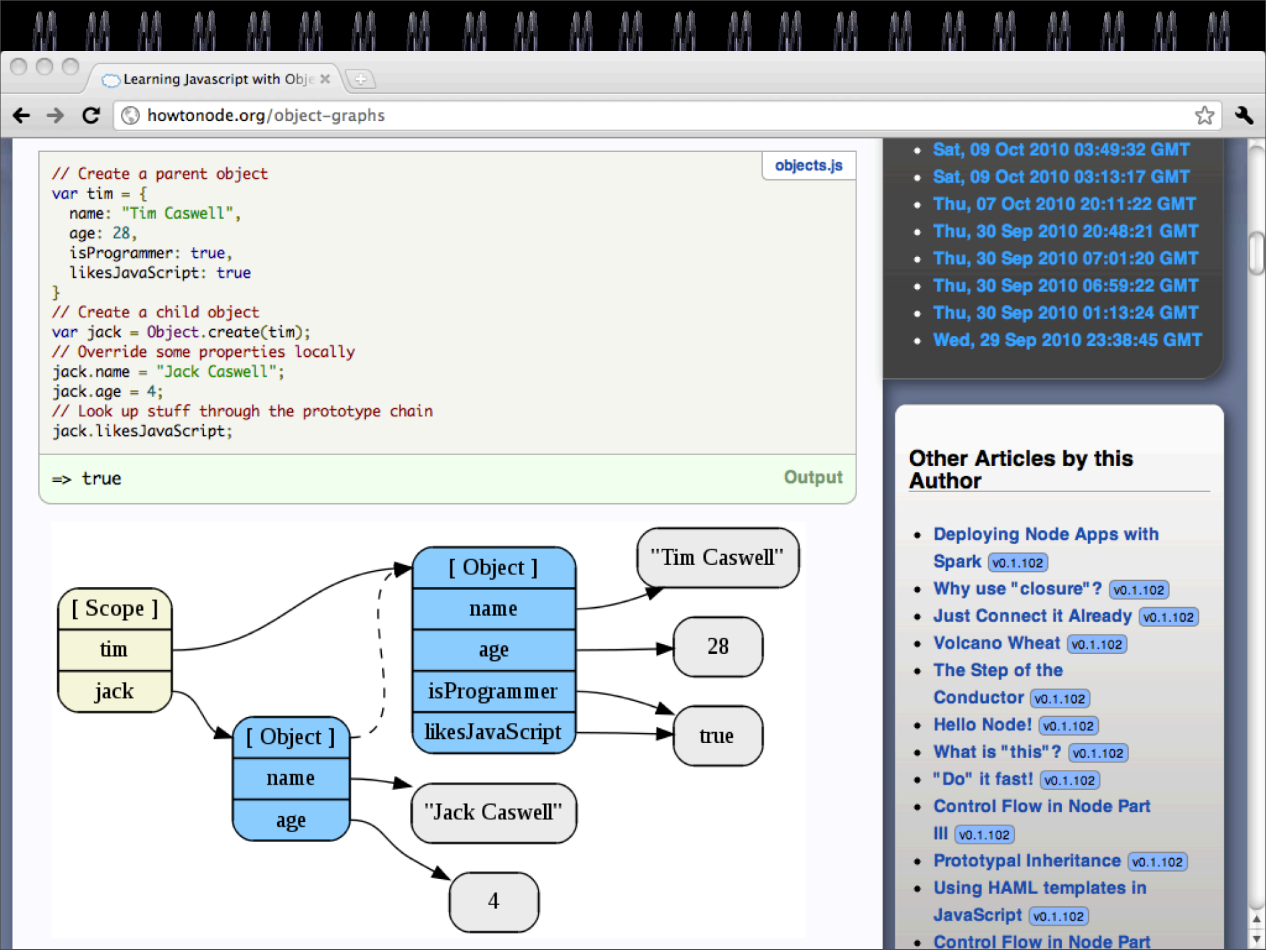
About the Author

Name:
Tim Caswell

Github:
creationix

Twitter:
creationix

Location:
Palo Alto, CA



Learning Javascript with Obje

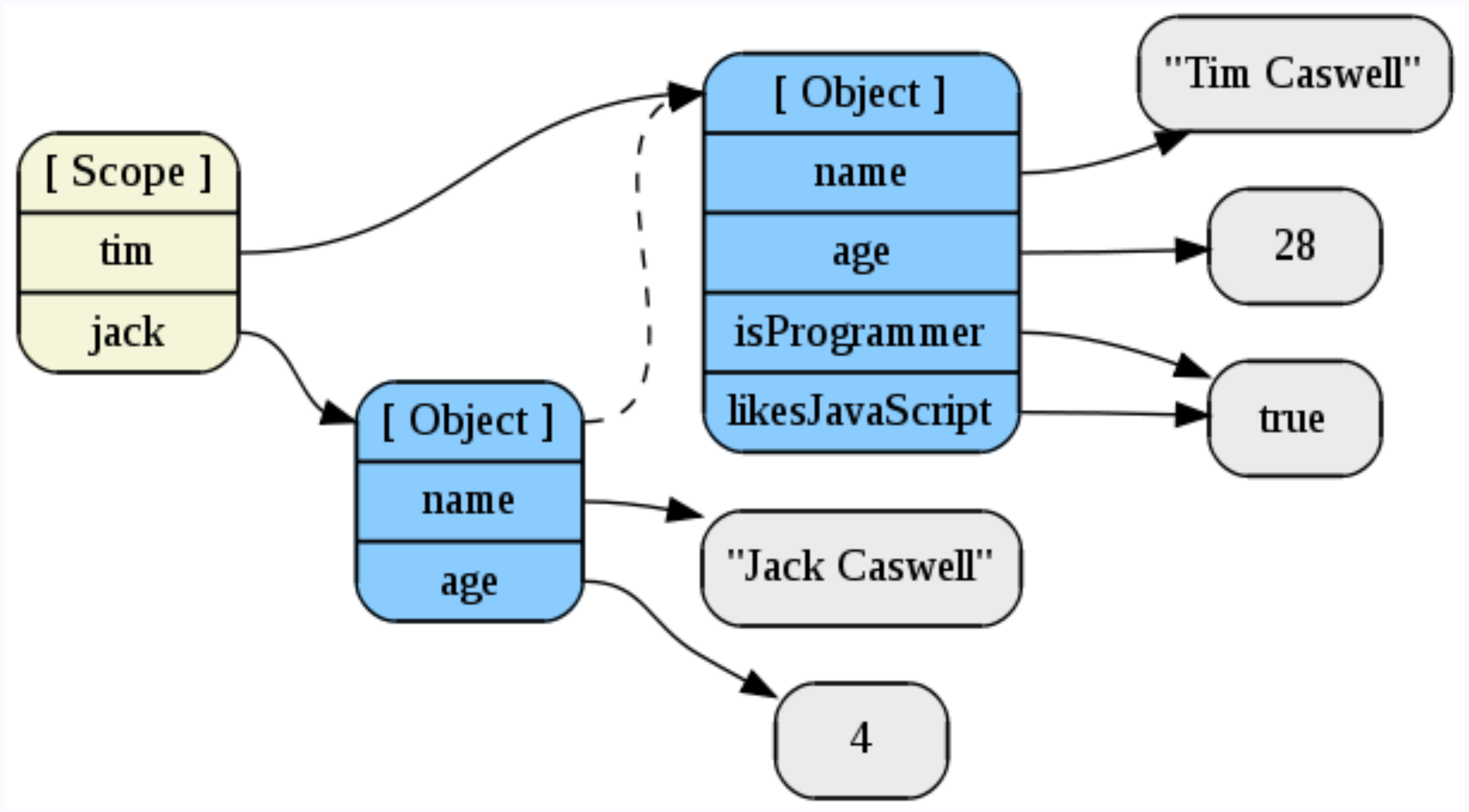
howtonode.org/object-graphs

objects.js

```
// Create a parent object
var tim = {
  name: "Tim Caswell",
  age: 28,
  isProgrammer: true,
  likesJavaScript: true
}
// Create a child object
var jack = Object.create(tim);
// Override some properties locally
jack.name = "Jack Caswell";
jack.age = 4;
// Look up stuff through the prototype chain
jack.likesJavaScript;
```

```
=> true
```

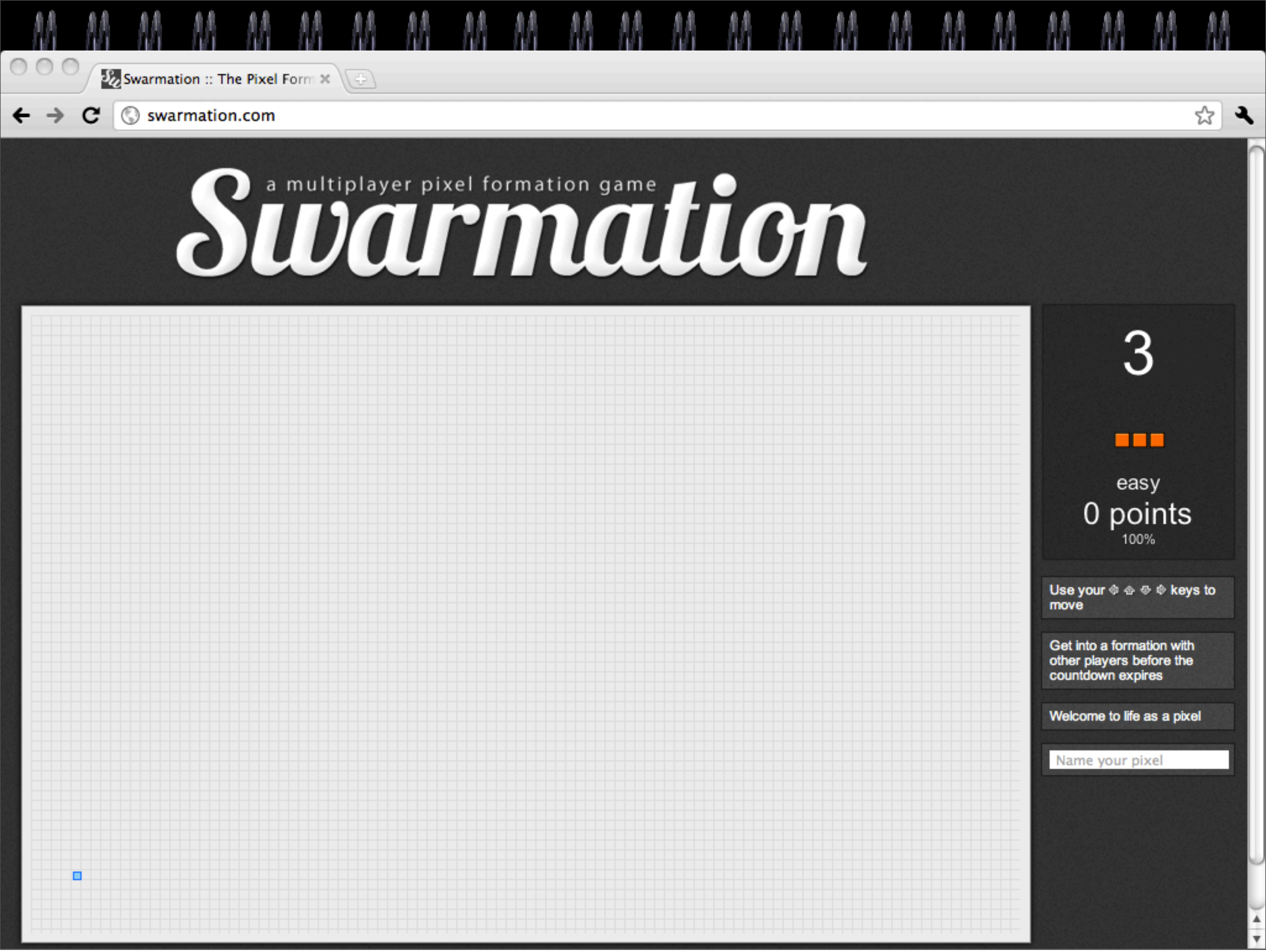
Output



- Sat, 09 Oct 2010 03:49:32 GMT
- Sat, 09 Oct 2010 03:13:17 GMT
- Thu, 07 Oct 2010 20:11:22 GMT
- Thu, 30 Sep 2010 20:48:21 GMT
- Thu, 30 Sep 2010 07:01:20 GMT
- Thu, 30 Sep 2010 06:59:22 GMT
- Thu, 30 Sep 2010 01:13:24 GMT
- Wed, 29 Sep 2010 23:38:45 GMT

Other Articles by this Author

- Deploying Node Apps with Spark v0.1.102
- Why use "closure"? v0.1.102
- Just Connect it Already v0.1.102
- Volcano Wheat v0.1.102
- The Step of the Conductor v0.1.102
- Hello Node! v0.1.102
- What is "this"? v0.1.102
- "Do" it fast! v0.1.102
- Control Flow in Node Part III v0.1.102
- Prototypal Inheritance v0.1.102
- Using HAML templates in JavaScript v0.1.102
- Control Flow in Node Part


Swarmation :: The Pixel Form
swarmation.com
a multiplayer pixel formation game
Swarmation

3

easy

0 points

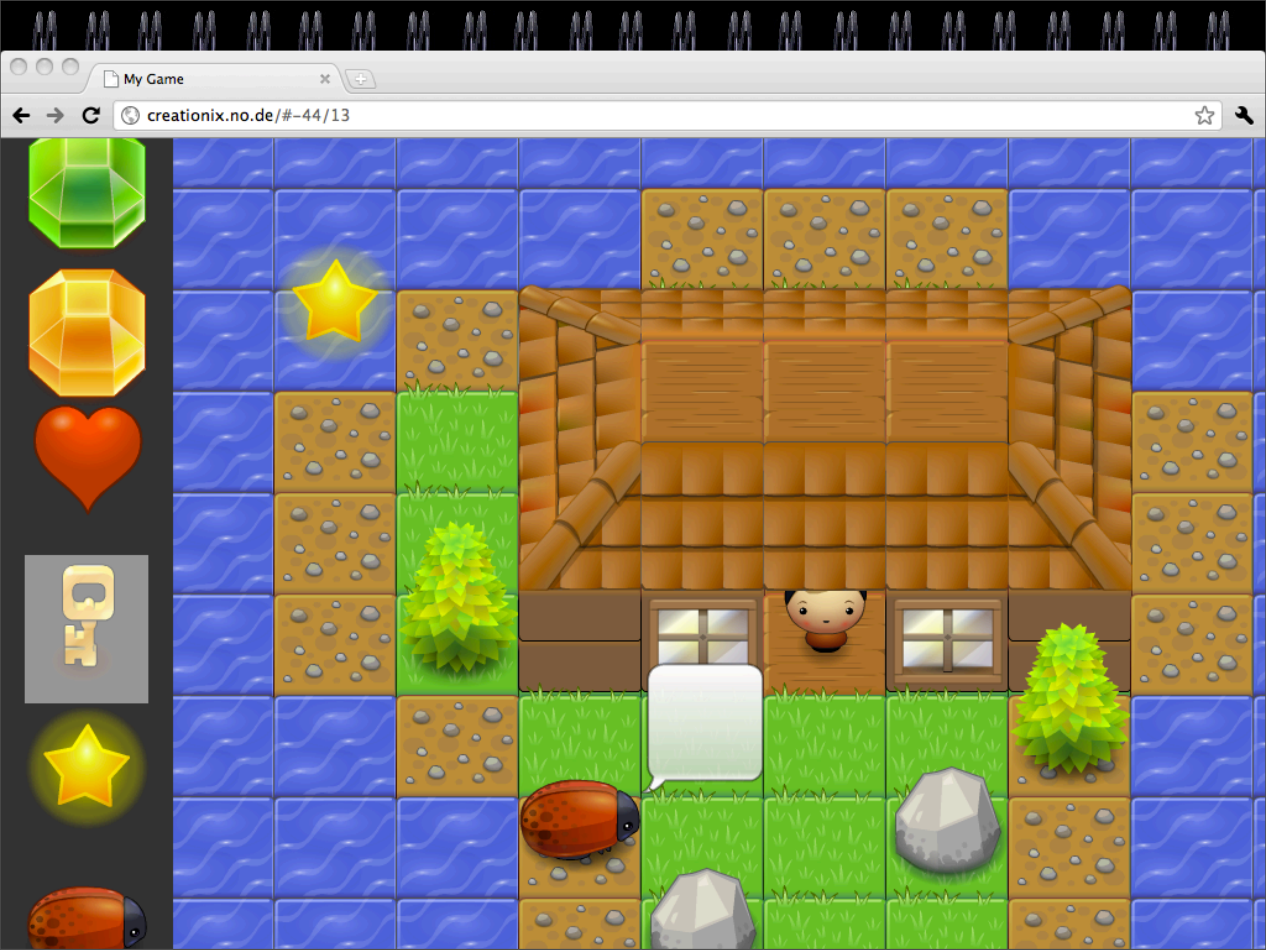
100%

Use your ← ↑ ↓ → keys to move

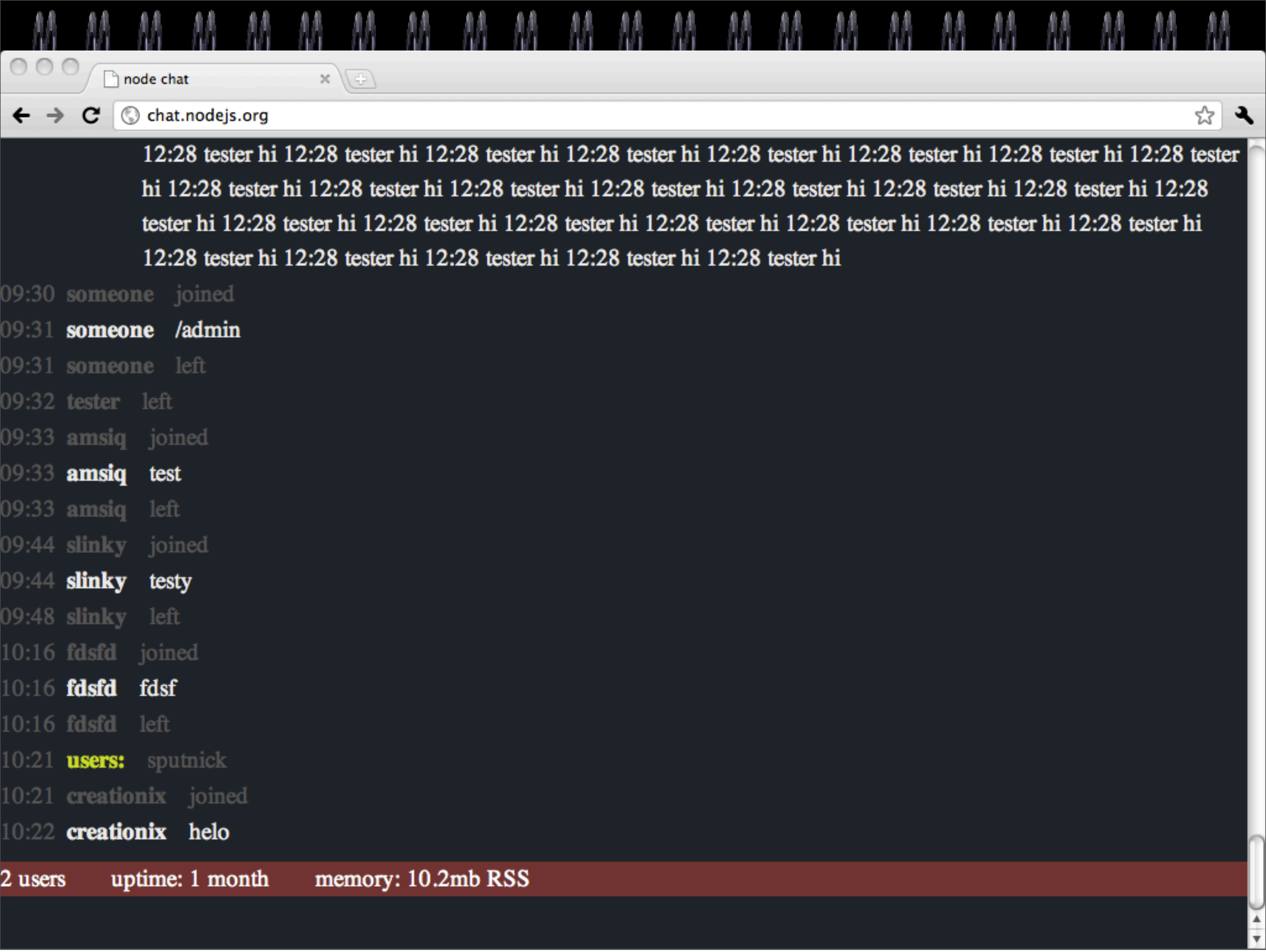
Get into a formation with other players before the countdown expires

Welcome to life as a pixel

Name your pixel



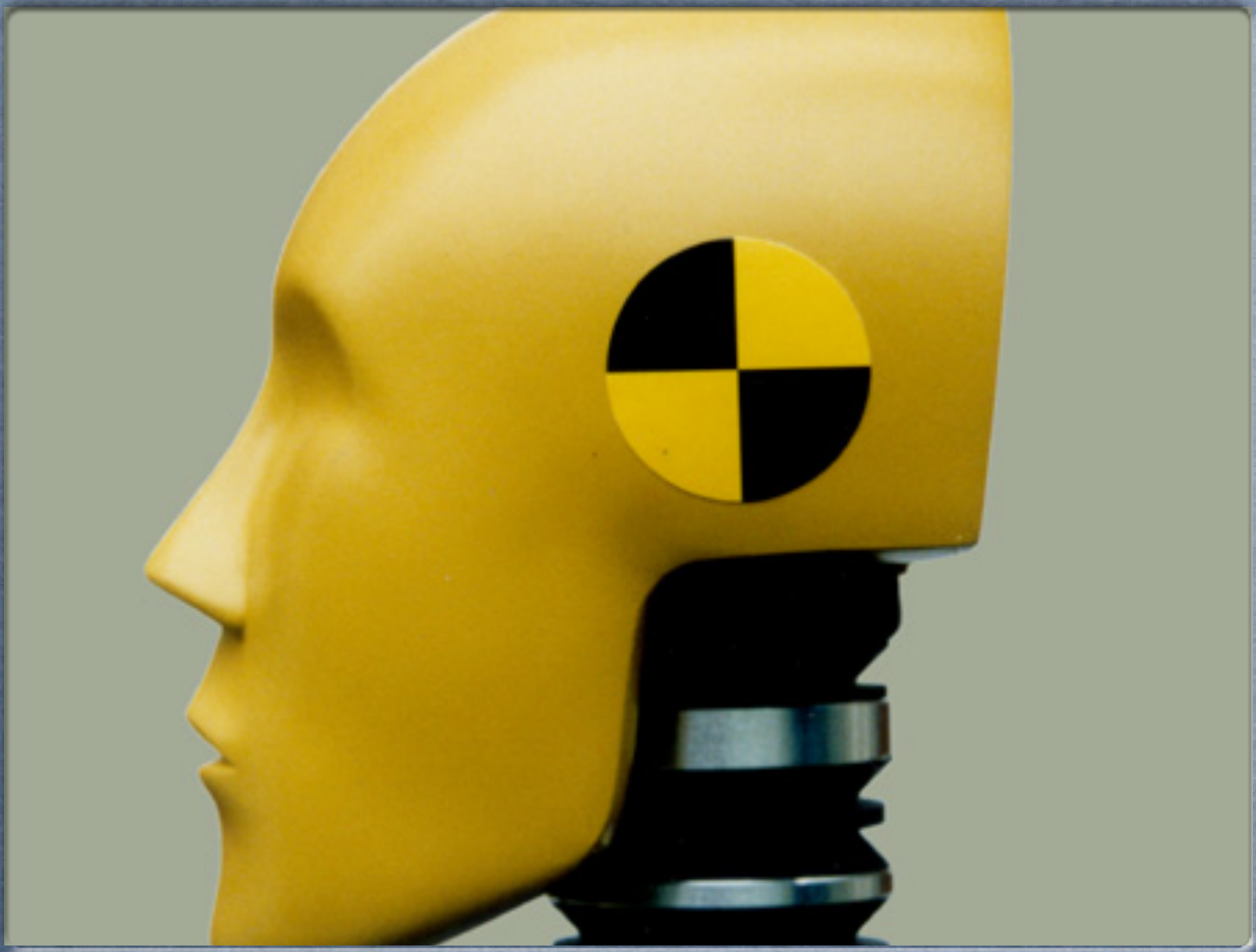
Saturday, October 9, 2010



node chat

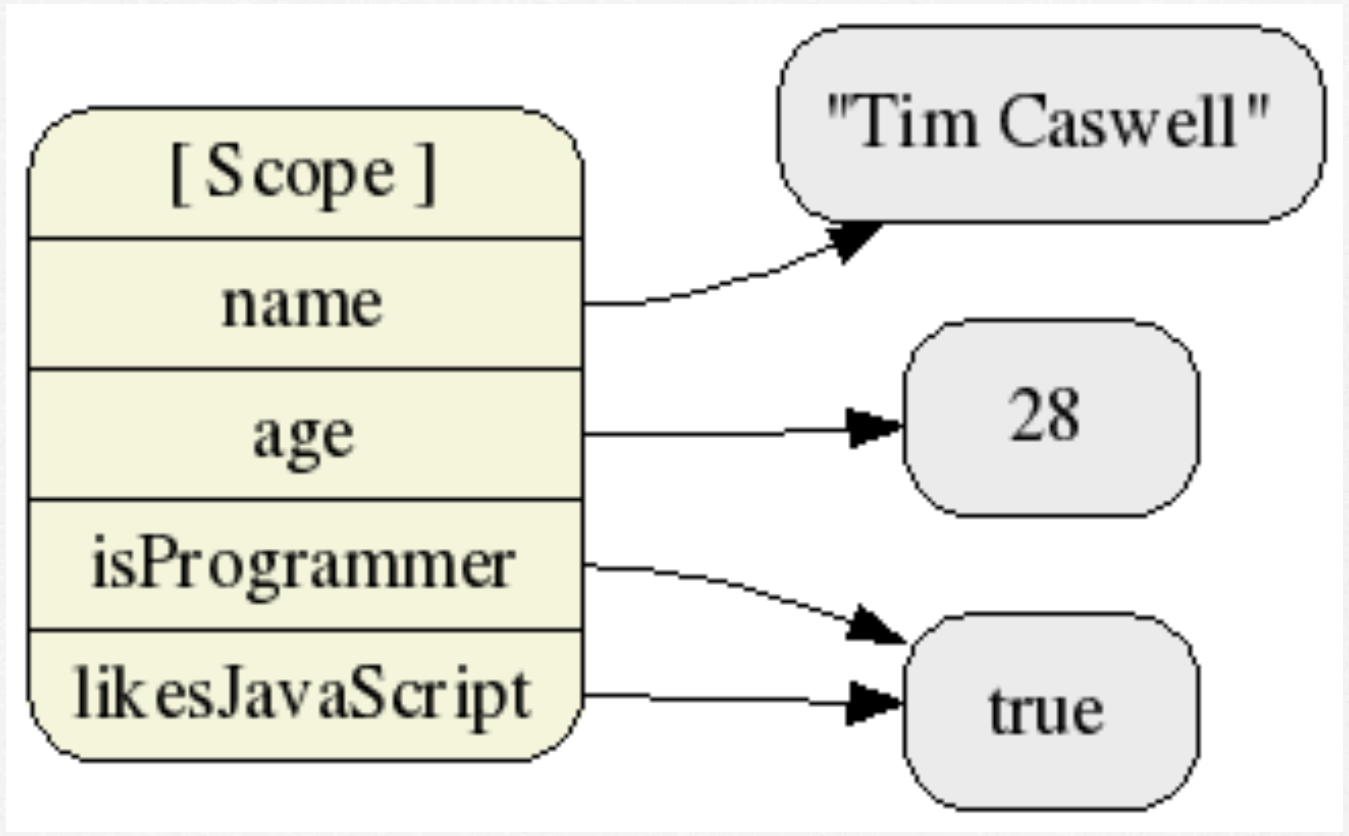
chat.nodejs.org

```
12:28 tester hi 12:28 tester hi 12:28 tester hi 12:28 tester hi 12:28 tester hi 12:28 tester hi 12:28 tester hi 12:28 tester
hi 12:28 tester hi 12:28 tester hi 12:28 tester hi 12:28 tester hi 12:28 tester hi 12:28 tester hi 12:28 tester hi 12:28
tester hi 12:28 tester hi 12:28 tester hi 12:28 tester hi 12:28 tester hi 12:28 tester hi 12:28 tester hi 12:28 tester hi
12:28 tester hi 12:28 tester hi 12:28 tester hi 12:28 tester hi 12:28 tester hi
09:30 someone    joined
09:31 someone    /admin
09:31 someone    left
09:32 tester   left
09:33 amsiq   joined
09:33 amsiq   test
09:33 amsiq   left
09:44 slinky   joined
09:44 slinky   testy
09:48 slinky   left
10:16 fdsfd   joined
10:16 fdsfd   fdsf
10:16 fdsfd   left
10:21 users:   sputnick
10:21 creationix   joined
10:22 creationix   helo
2 users      uptime: 1 month      memory: 10.2mb RSS
```

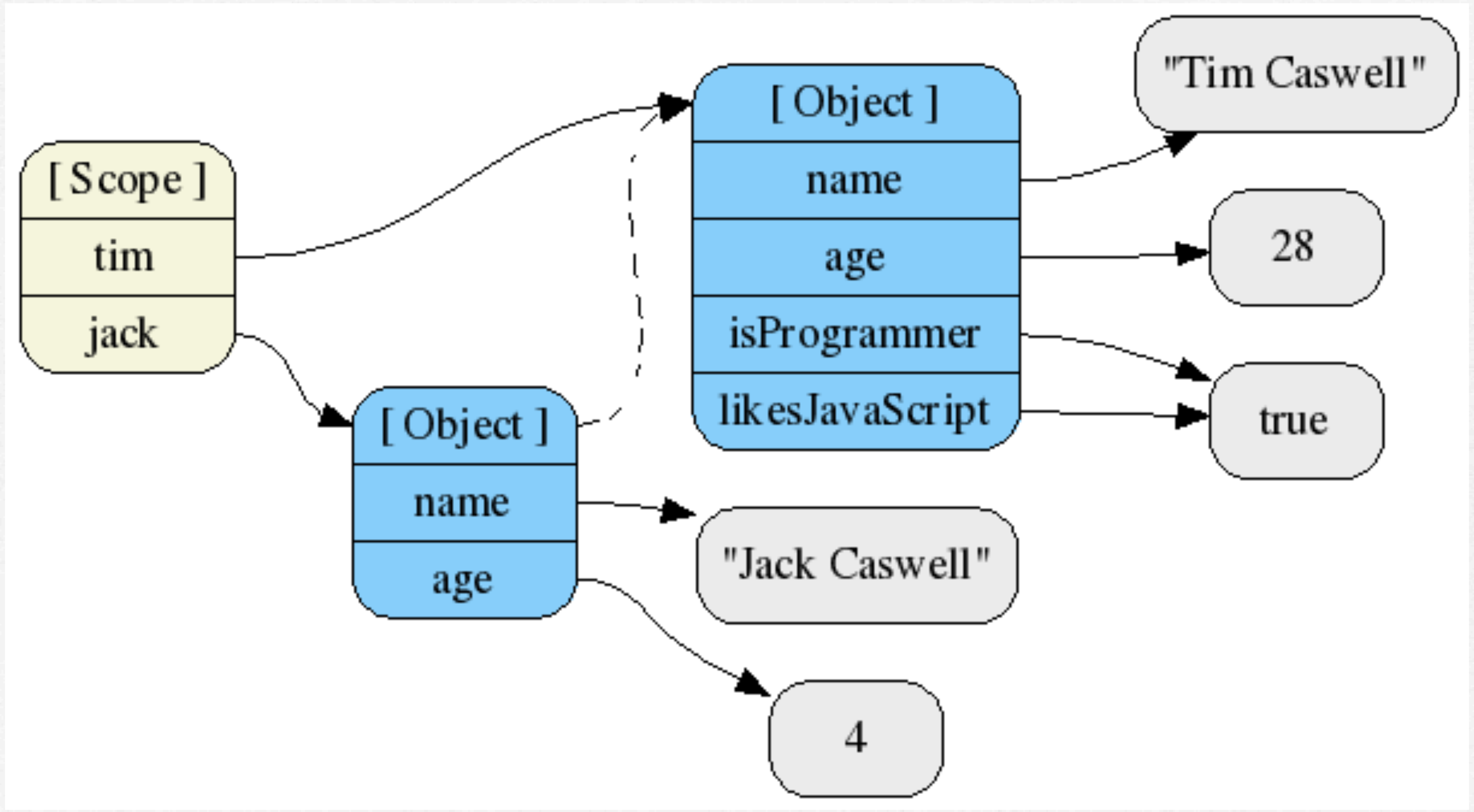


Advanced JavaScript Crash Course

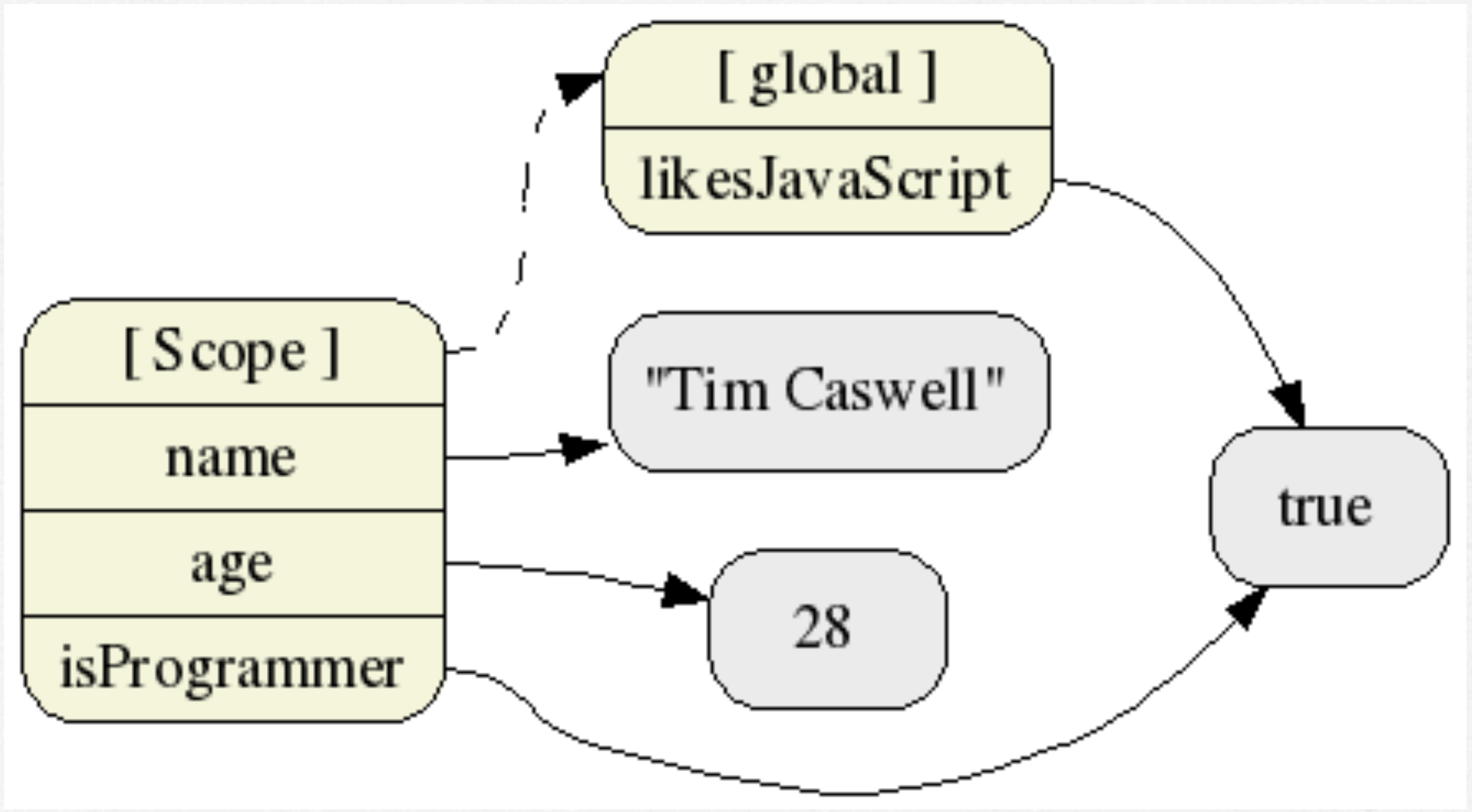
Free Variables



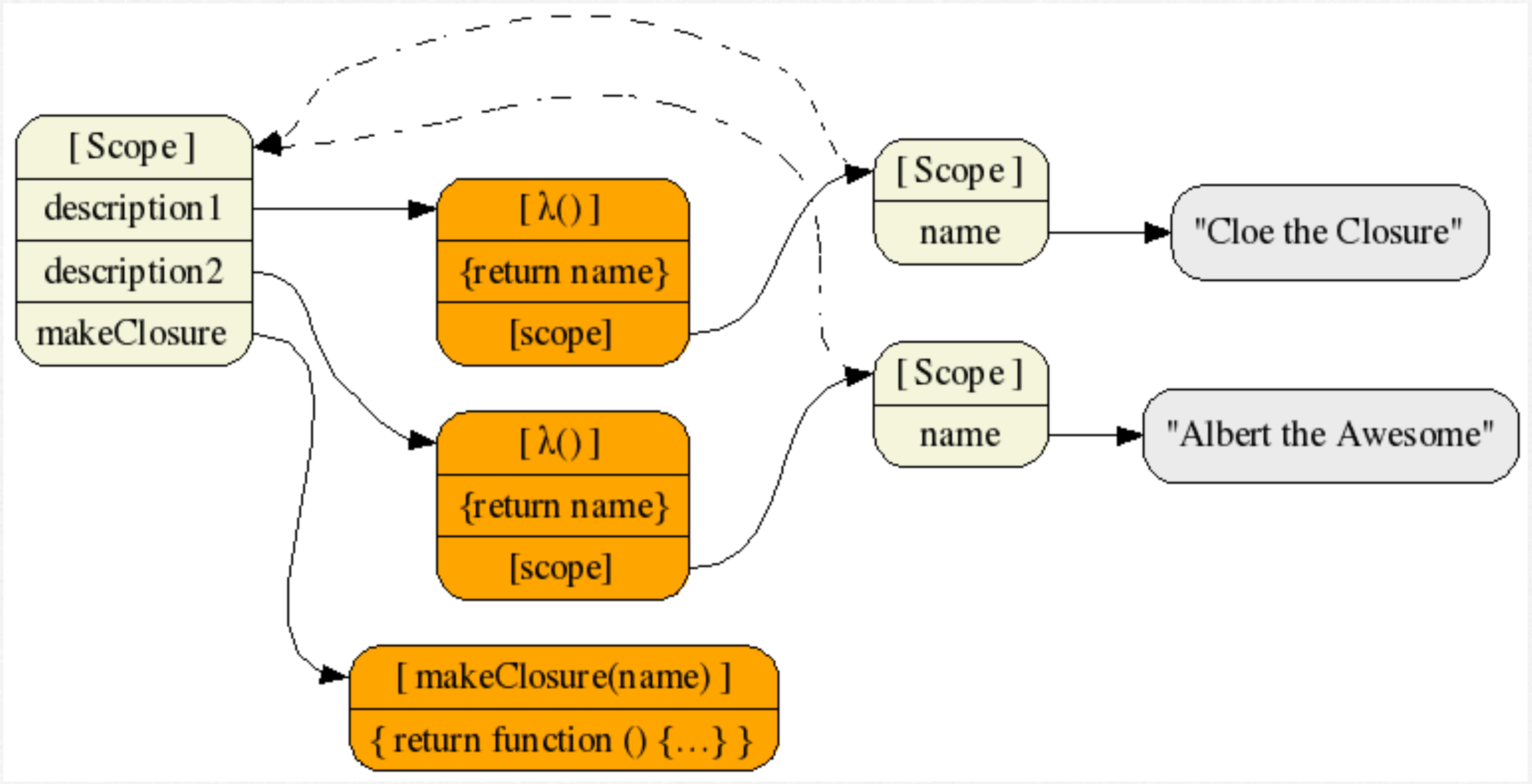
Object Inheritance



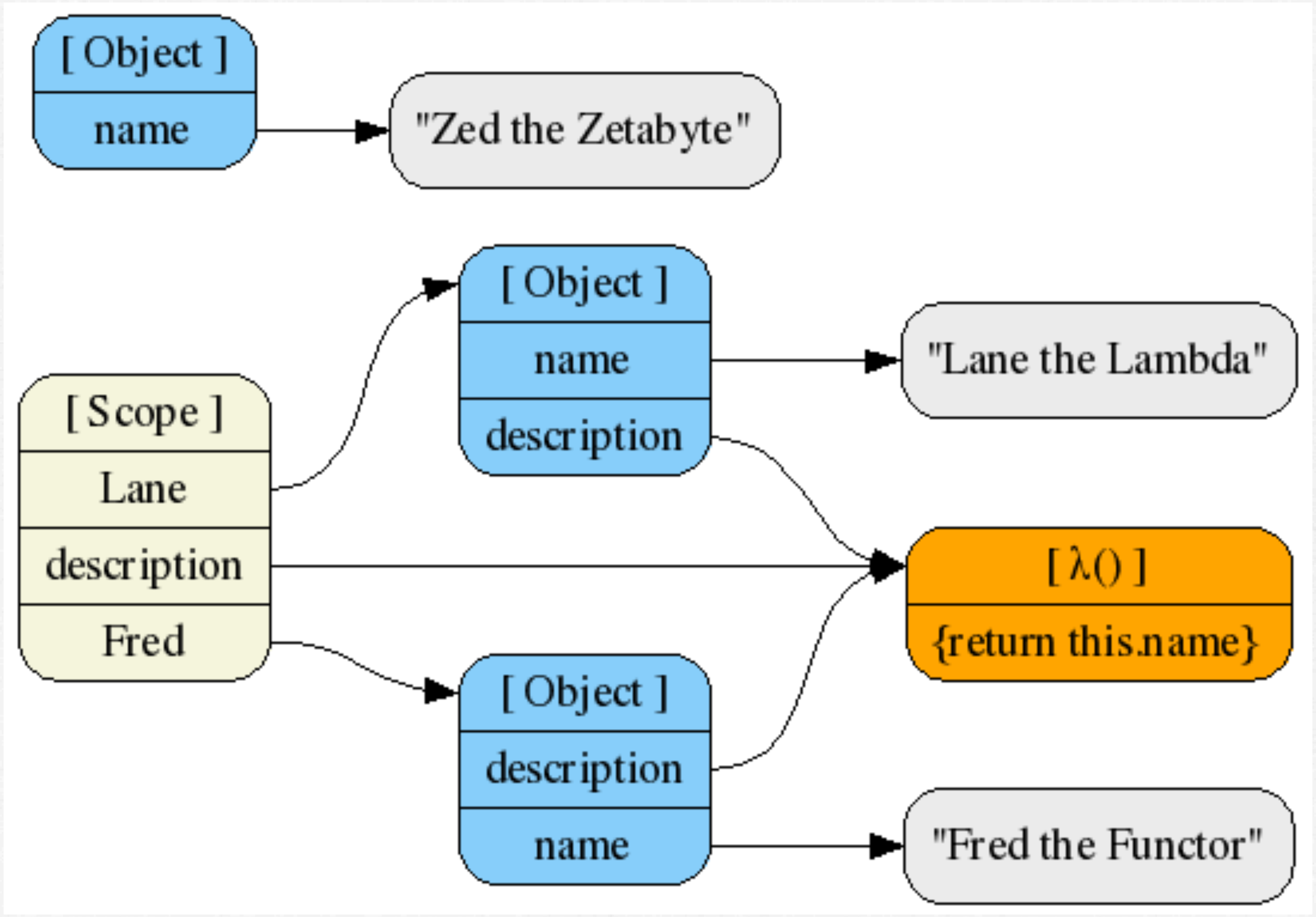
Global Variables



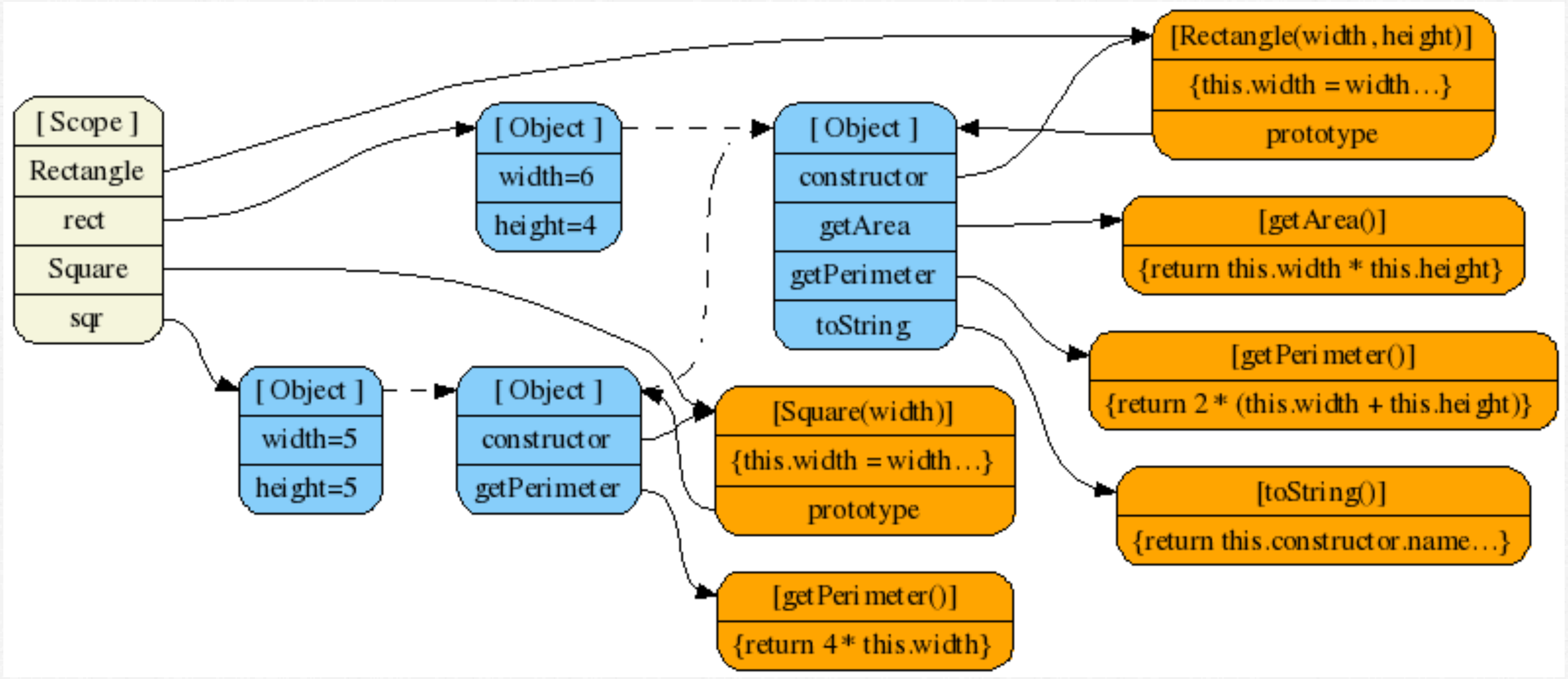
Closures


[Scope]
description1
description2
makeClosure
[λ()]
{return name}
[scope]
[λ()]
{return name}
[scope]
[makeClosure(name)]
{ return function () {…} }
[Scope]
name
"Cloe the Closure"
[Scope]
name
"Albert the Awesome"

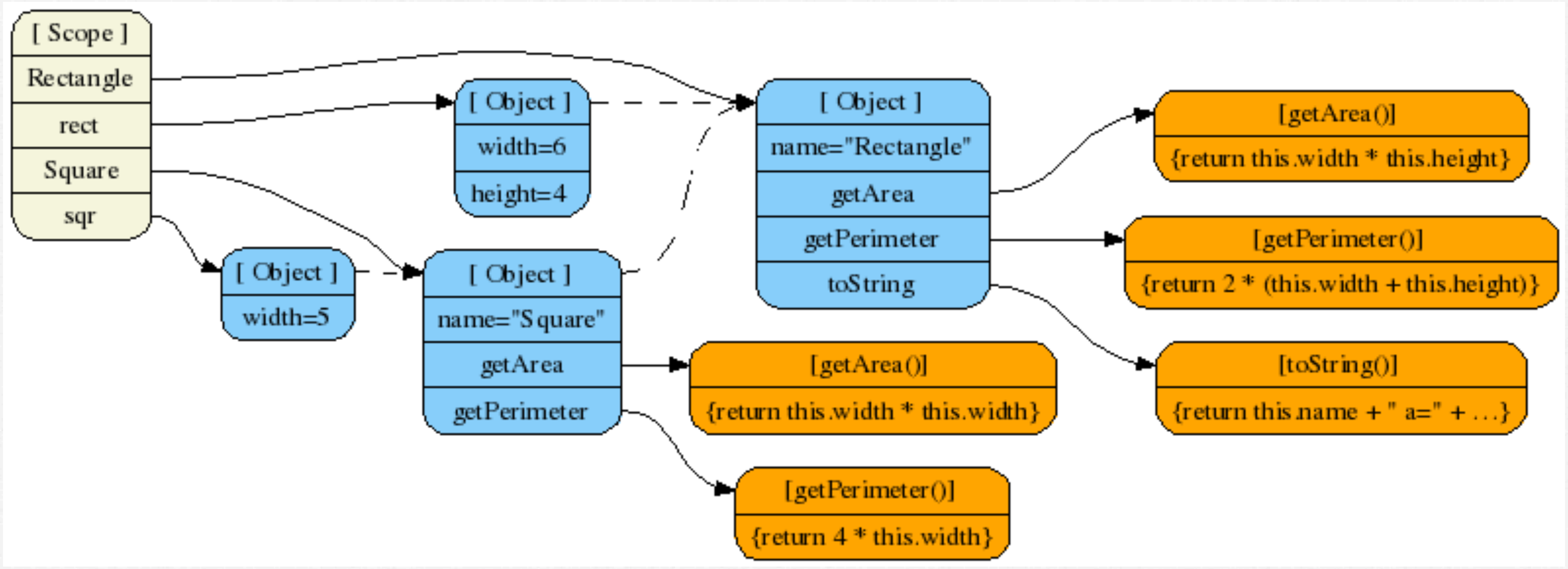
Shared Functions



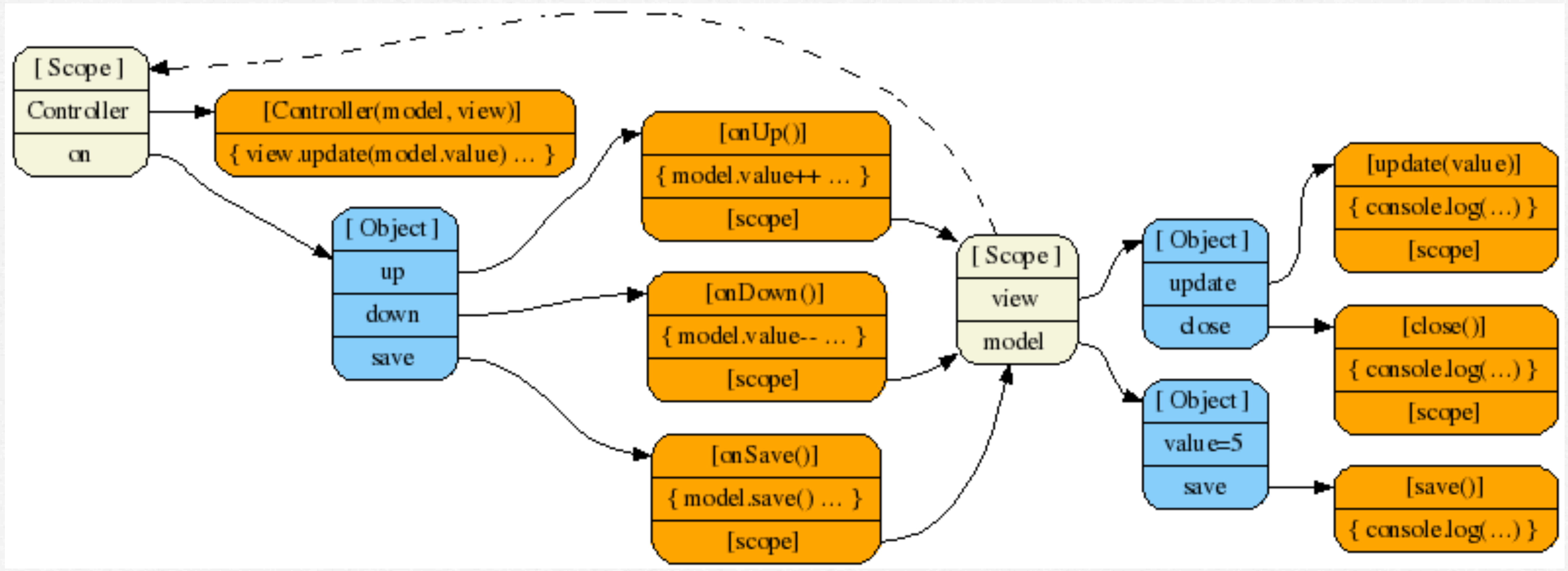
Object Constructors



Prototypal Inheritance



Factory Pattern



Live Coding Time!

Questions?

Tim Caswell
tim@creationix.com
http://howtonode.org/