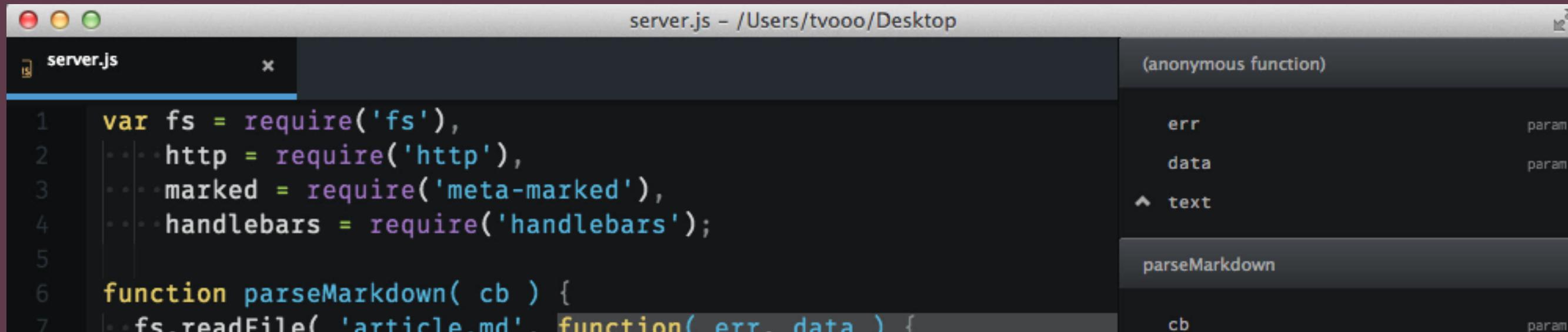


# Making Scope Explorable in Software Development Environments

---

by *Tim von Oldenburg*

---



A screenshot of a software development environment, likely an IDE or browser-based editor, showing a file named 'server.js'. The code is written in JavaScript and defines a function 'parseMarkdown' that reads a file from the 'article.md' path. A code completion dropdown menu is open at the bottom right of the screen, listing parameters for the 'parseMarkdown' function: 'err', 'data', and 'cb'. The 'err' parameter is labeled as a 'param'. The 'data' parameter is also labeled as a 'param'. The 'cb' parameter is also labeled as a 'param'. The 'text' parameter is preceded by an upward arrowhead, indicating it is a local variable or parameter that has already been defined.

```
server.js - /Users/tvooo/Desktop
server.js

1 var fs = require('fs'),
2     http = require('http'),
3     marked = require('meta-marked'),
4     handlebars = require('handlebars');
5
6 function parseMarkdown( cb ) {
7   fs.readFile( 'article.md', function( err, data ) {
```

(anonymous function)

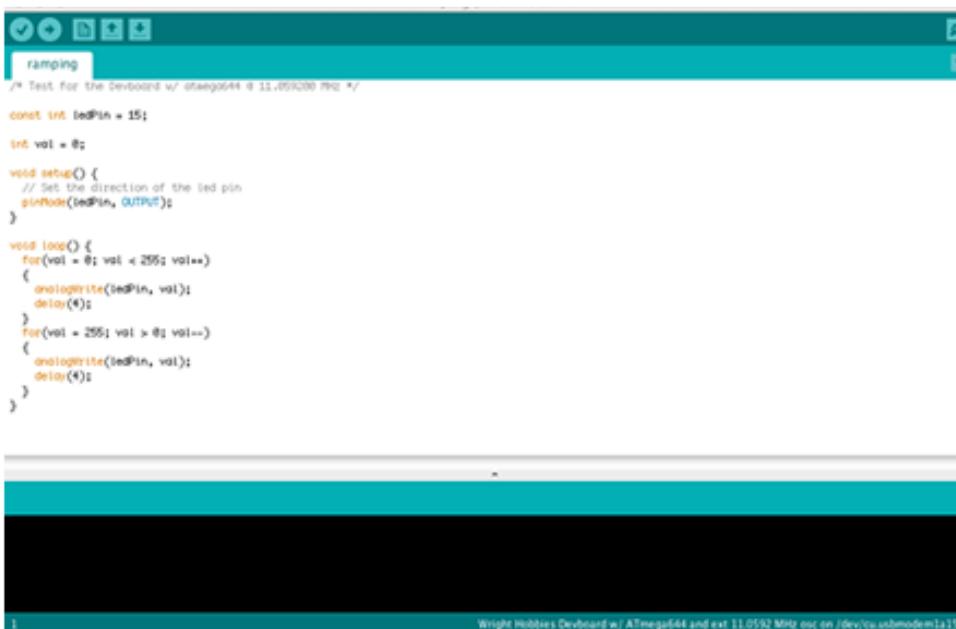
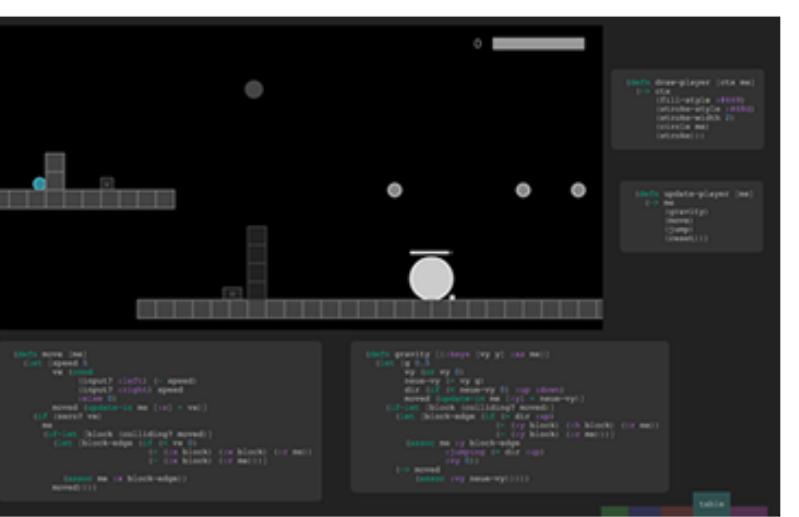
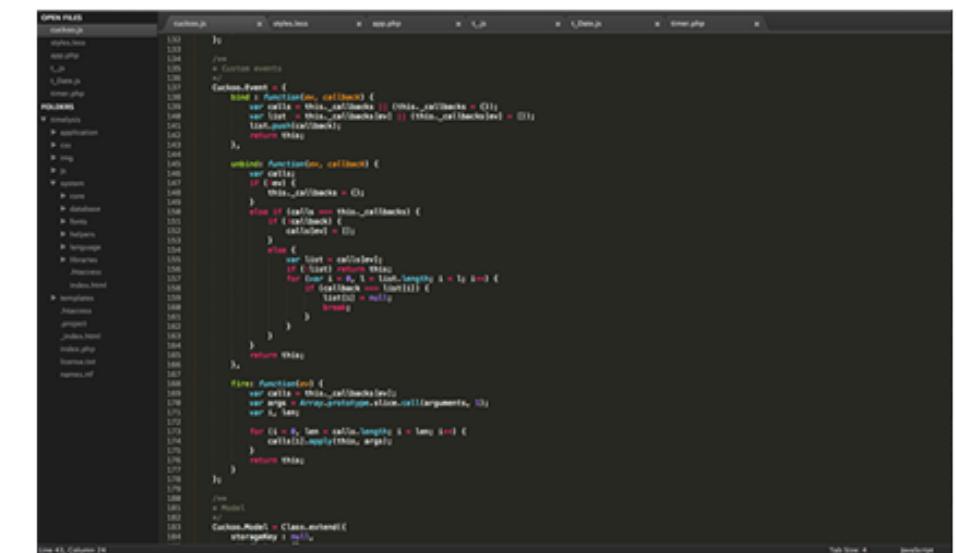
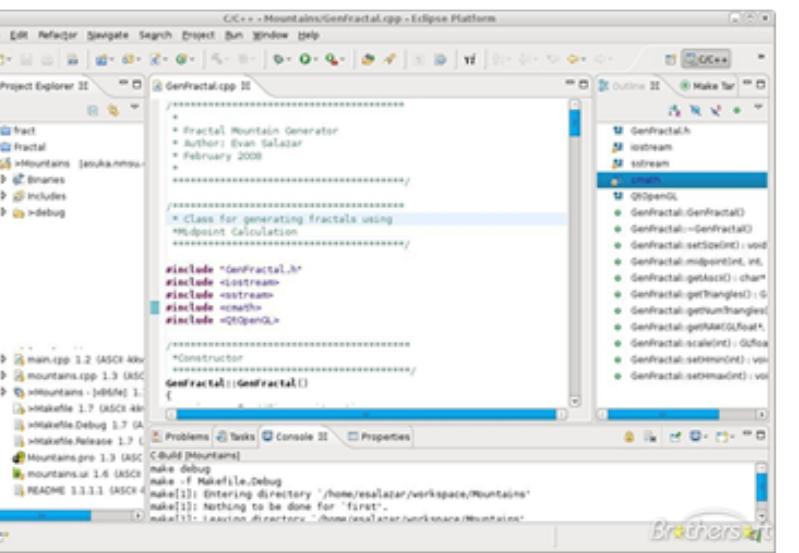
- err param
- data param
- ▲ text
- parseMarkdown
- cb param

# Integrated Development Environments (IDEs)

IDEs integrate a code editor with different tools for software development:

- workflow
- building and distributing
- collaboration
- quality assurance
- ...

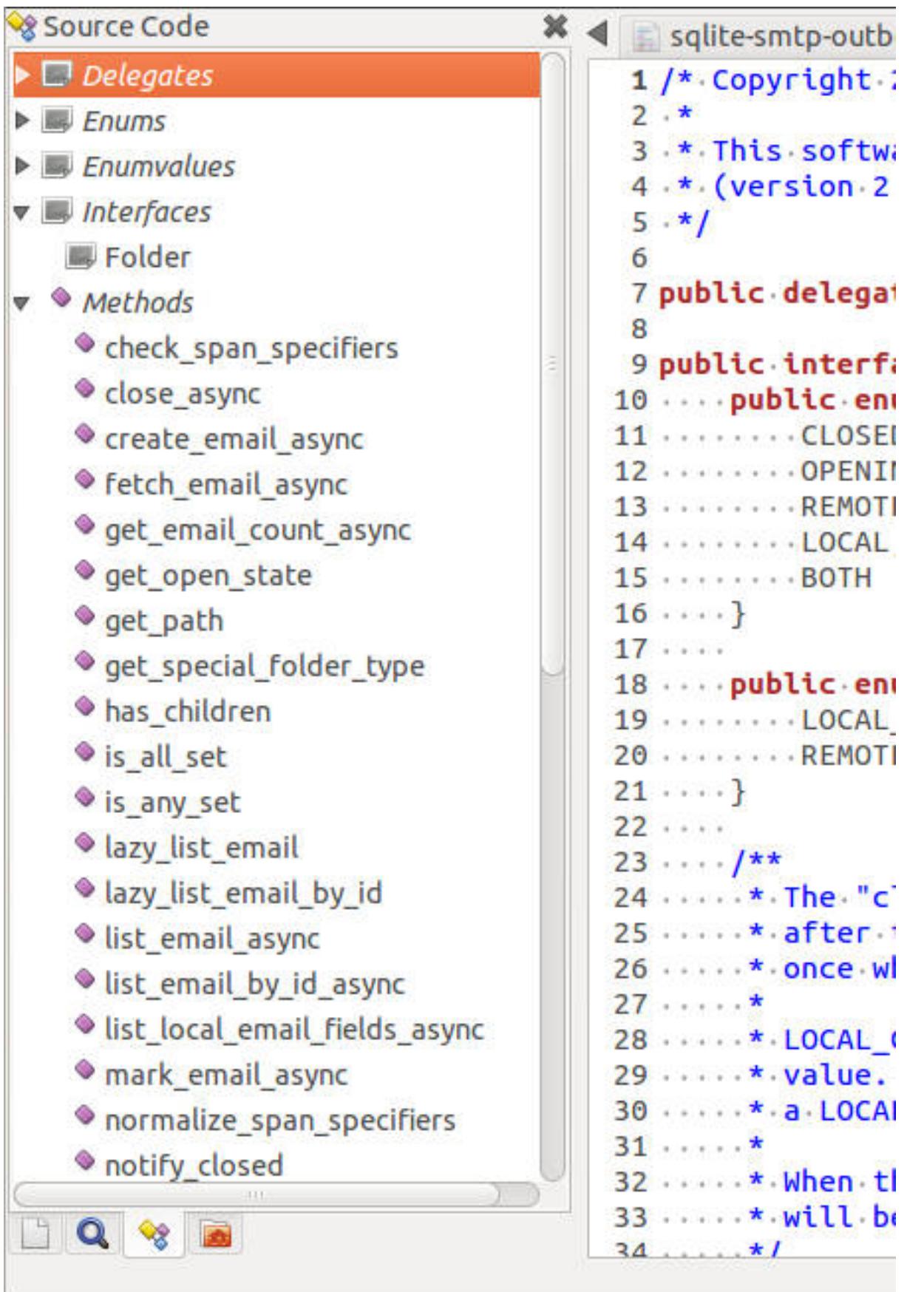
# IDE Examples



## Syntax Highlighting

```
1 <!DOCTYPE html PUBLIC "-//W3C/DTD HTML
2 <html>
3   <head>
4     <title>Example</title>
5     <link href="screen.css" rel="sty
6   </head>
7   <body>
8     <h1>
9       <a href="/">Header</a>
10    </h1>
11    <ul id="nav">
12      <li>
13        <a href="one/">One</a>
14      </li>
15      <li>
16        <a href="two/">Two</a>
17      </li>
```

Visually distinguish language elements.

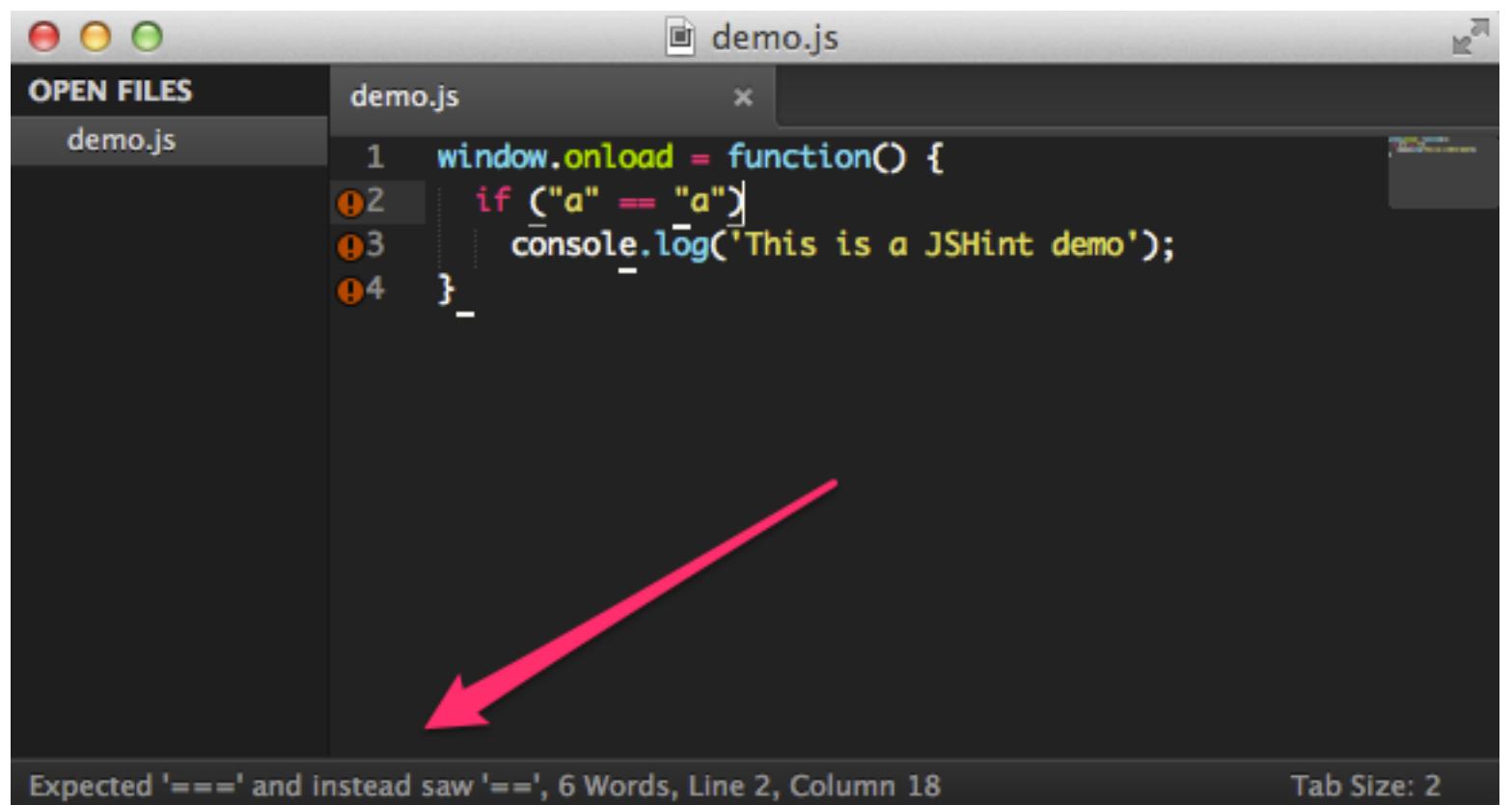


# Class Browser

Browse language symbols, like variables and methods.

## Linter

Find code smells (bad style).



```
demo.js
1 window.onload = function() {
2     if ("a" == "a")
3         console.log('This is a JSHint demo');
4 }
```

Expected '===' and instead saw '==', 6 Words, Line 2, Column 18

Tab Size: 2

# Scope

is „*the set of rules that determines where and how a variable (identifier) can be looked-up*“ (Simpson 2014)

Different languages, different scoping models.

# JavaScript

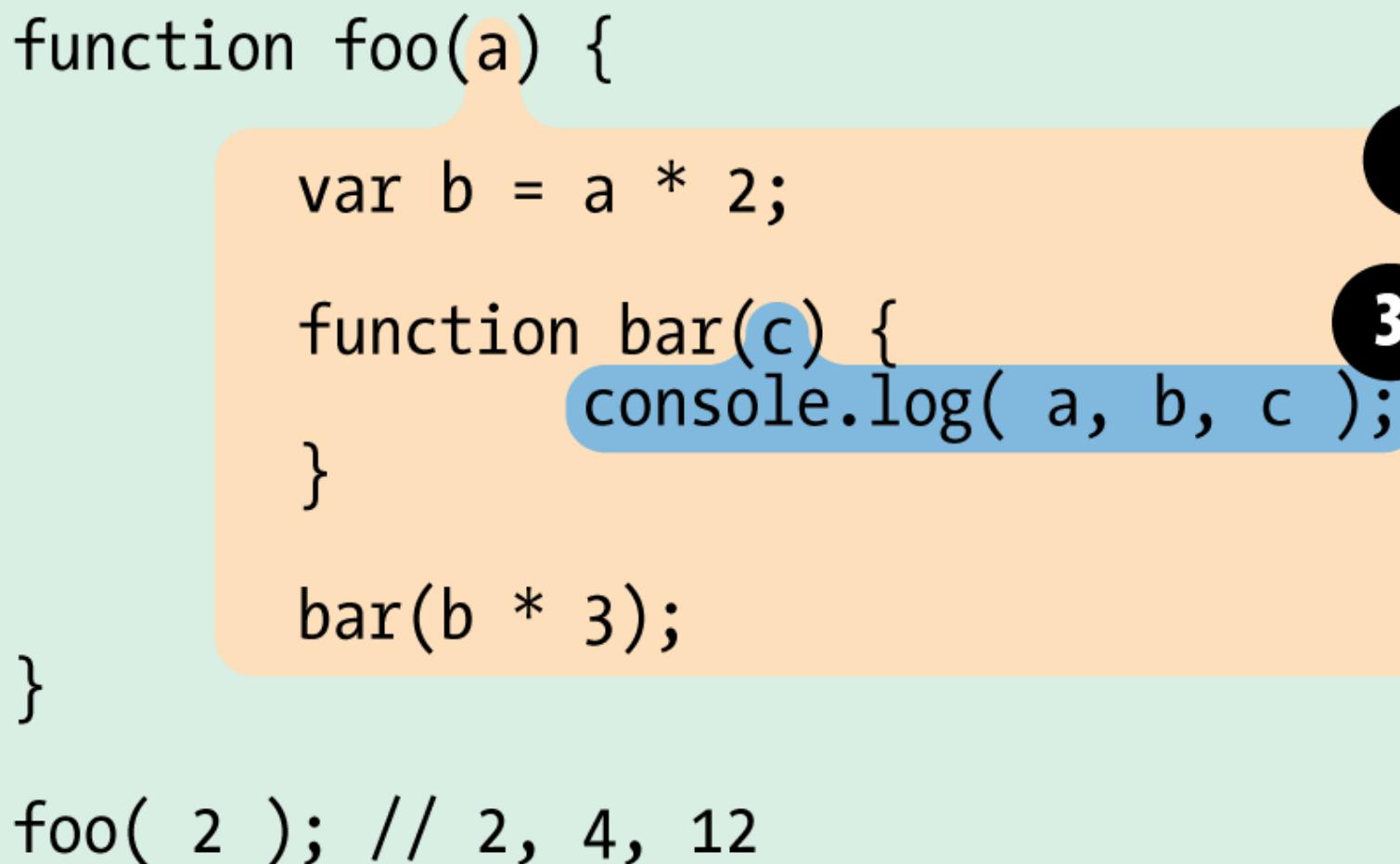
---

The language of the world wide web.

# Scope in JavaScript

## Nested function scope

```
function foo(a) {  
    var b = a * 2;  
  
    function bar(c) {  
        console.log( a, b, c );  
    }  
  
    bar(b * 3);  
}  
  
foo( 2 ); // 2, 4, 12
```



The diagram illustrates the scope chain for the provided JavaScript code. It uses colored boxes to represent the scopes:

- Outer Scope (Green):** Contains the declaration of the `foo` function.
- Inner Scope (Orange):** Contains the declaration of the `bar` function and its body.
- Lexical Scope (Blue):** Contains the `console.log` statement within the `bar` function body.

Three numbered callouts point to specific parts of the code:

- 1:** Points to the opening brace of the `foo` function.
- 2:** Points to the `var` declaration inside the `foo` function.
- 3:** Points to the `console.log` statement inside the `bar` function body.

## Phenomena/Problems with Scope in JavaScript

- Shadowing
- Hoisting
- Performance
- Closure

Goal: making the user aware of those phenomena in their code

Introducing

---

# JavaScript Scope Inspector

server.js - /Users/tvooo/Desktop

```
server.js      x

1  var fs = require('fs'),
2  http = require('http'),
3  marked = require('meta-marked'),
4  handlebars = require('handlebars');
5
6  function parseMarkdown( cb ) {
7    fs.readFile( 'article.md', function( err, data ) {
8      if ( err ) {
9        return cb( err );
10     }
11    var text = marked( data.toString() );
12    cb( null, text );
13  });
14}
15
16 function compileHtml( text, cb ) {
17  fs.readFile( 'template.html', function( err, data ) {
18    if ( err ) {
19      return cb( err );
20    }
21    var template = handlebars.compile( data.toString() );
22    cb( null, template({
23      meta: text.meta,
24      content: text.html
25    }) );
26  });
27}
28
29 var server = http.createServer( function( req, res ) {
30   parseMarkdown( function( err, data ) {
```

(anonymous function)

- err param
- data param
- text
  - parseMarkdown

cb param

(anonymous function) ()

GLOBAL

- fs
- http
- marked
- handlebars
- server
  - parseMarkdown ()
  - compileHtml ()
  - (anonymous function) ()

GLOBAL > parseMarkdown > (anonymous function)

# Process

User-Centered Design Process applied to the Open Source community.

- Exploration & Research
- Ideation
- Prototyping
- Testing

## **Survey**

42 answers in 2 weeks

*Which IDE features do you use, and how would you make them better?*

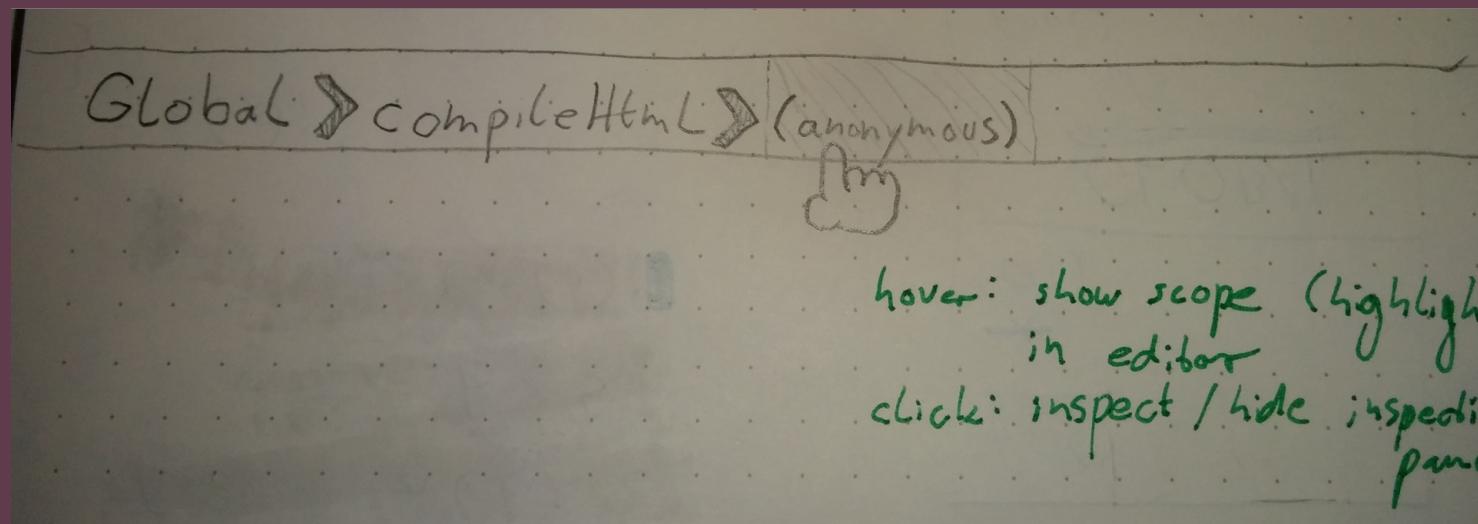
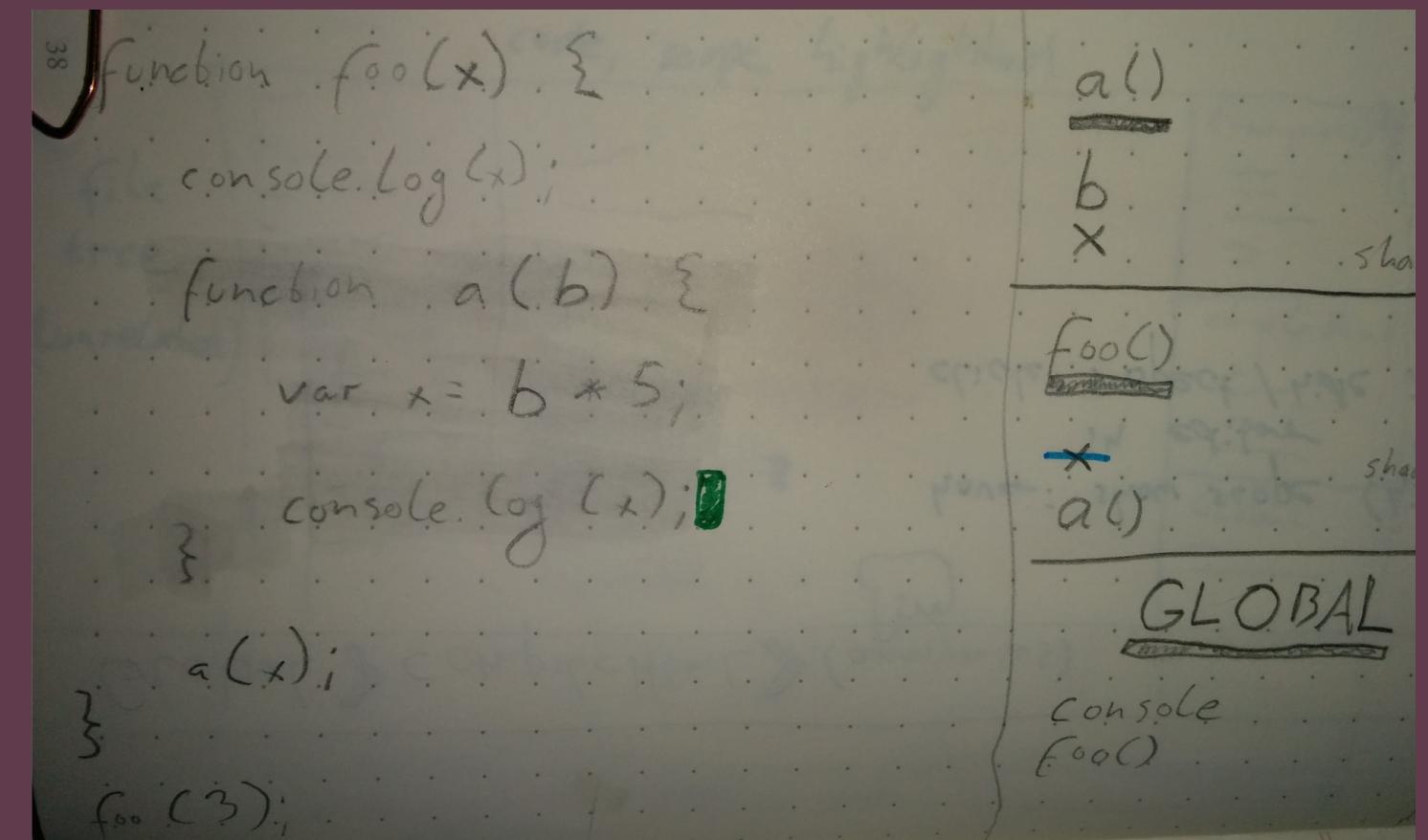
## Interviews

7 participants, about 10 hours of interview time

## Result

- Scope
- Characteristics of well-integrated language tools
  - Performance
  - Modularity
  - Smartness
  - Focus on Code

# Sketches



# Scripted Prototype

---

The screenshot shows a code editor window titled "Document". On the left, there is a block of JavaScript code. On the right, there is a sidebar with three tabs: "canon", "parseMarkdown()", and "Global".

**canon**

		arg
err		arg
data		arg
text		

**parseMarkdown()**

	arg
cb	arg

**Global**

fs
http
marked
handlebars
server
parseMarkdown
compileHtml

```
var fs = require('fs'),
    http = require('http'),
    marked = require('meta-marked'),
    handlebars = require('handlebars');

function parseMarkdown( cb ) {
  fs.readFile( 'article.md', function( err, data ) {
    if ( err ) {
      return cb( err );
    }

    var text = marked( data.toString() );
    cb( null, text );
  });
}

function compileHtml( text, cb ) {
  fs.readFile( 'template.html', function( err, data ) {
    if ( err ) {
      return cb( err );
    }
  });
}
```

# Working Prototype

---

The screenshot shows a Mac OS X terminal window with a dark theme. The title bar reads "server.js - /Users/tvooo/Desktop". The main pane contains the following JavaScript code:

```
1 var fs = require('fs'),
2     http = require('http'),
3     marked = require('meta-marked'),
4     handlebars = require('handlebars');
5
6 function parseMarkdown( cb ) {
7   fs.readFile( 'article.md', function( err, data ) {
8     if ( err ) {
9       return cb( err );
10    }
11    var text = marked( data.toString() );
12    cb( null, text );
13  });
14}
15
16 function compileHtml( text, cb ) {
17   fs.readFile( 'template.html', function( err, data ) {
18     if ( err ) {
19       return cb( err );
20     }
21     var template = handlebars.compile( data.toString() );
22     cb( null, template({
```

To the right of the code editor is a sidebar titled "(anonymous function)". It lists several function definitions with their parameters:

- err (param)
- data (param)
- text (param)
- parseMarkdown (0)
- cb (param)
- (anonymous function) (0)

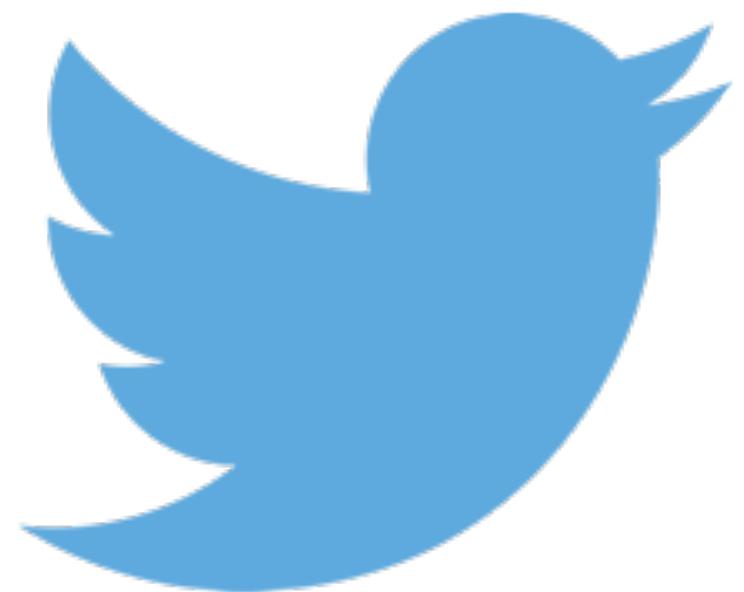
Below these, under the "GLOBAL" section, are the module exports:

- fs
- http
- marked
- handlebars
- server

Under the "server" entry, the functions listed are:

- parseMarkdown (0)
- compileHtml (0)
- (anonymous function) (0)

# Test installment



## **Test installment problems**

The scheduled test users where not available for week-long testing.

**Instead: Remote contextual inquiry.**

## Testing results (qualitative)

*“Why can’t I navigate using the sidebar? Jump to variables?”*

*“I want to enable/disable the bottom bar and the highlighting.”*

*“It doesn’t get updated when I type.”*

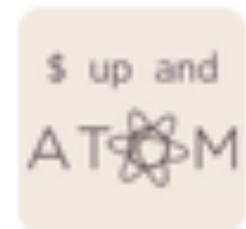
(paraphrased)

## Testing results (quantitative)

After 1 week of tracking metrics:

**150 downloads – 5 users tracking**

## Social Media Feedback



**Atom Tips**

@AtomTips



Following

Really like the look of JS Scope Inspector  
by @sweeneytimm:

[atom.io/packages/scope...](https://atom.io/packages/scope)

## Social Media Feedback



**Reginald Braithwaite**

@raganwald



Following

How cool is this?

[atom.io/packages/scope...](http://atom.io/packages/scope...)

## Social Media Feedback

adardesign  
@adardesign



+ Follow

github's #atom scope-inspector, another reason to consider switching from #sublimetext [bit.ly/1mXUpEf](http://bit.ly/1mXUpEf)

## Reflection (1)

- Disabling **tracking by default** was hindering
- **Testing** in the open source community **is hard**, especially with a narrow user group
- Using a **small-yet-growing platform** might not be ideal for prototyping
- **Social Media** are suitable channels for “marketing and communications”

## Reflection (2)

“Every good work of software starts by scratching a developer's personal itch.” (Raymond 1999, paraphrased)

Interaction Design offers new impulses for the technology-driven area of development software—by *scratching other developer's itches*.

## Contributions

- Four Characteristics of well-integrated language tools: *Performance, Modularity, Smartness, Focus on Code*
- A case study of evaluating a design with a specific, narrow target group
- Working prototype (open source)
- Library for scope analysis (open source)

# Thank you!

---

# Post-Submit

---

What happened in the past three months

## What happened in the past three months

- Minor bug fixes
- Attended to user testing feedback
  1. navigation through sidebar
  2. enable everything separately
  3. re-evaluate while typing
- Context colouring as separate plugin by third party

What happened in the past three months

Nearing 400 downloads

**scope-inspector**

Inspect and explore JavaScript scope.

 tvooo

Last updated July 27, 2014  367  7

What happened in the past three months

Enabled GA tracking by default



The sidebar is enabled on demand; Breadcrumbs are usually enabled.