Syvis

A visual code editor for Lively

Software Architecture Group

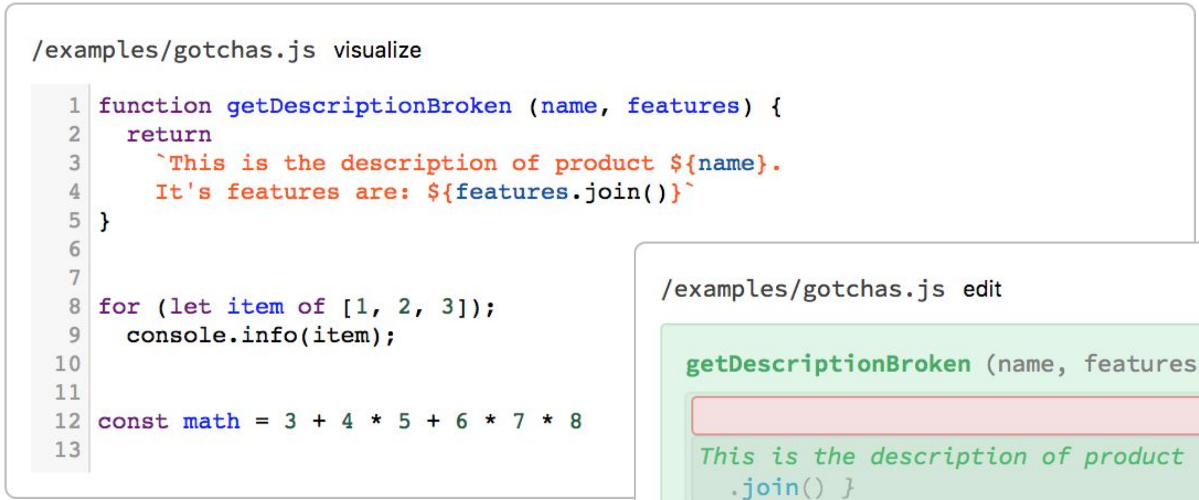
Prof. Dr. Robert Hirschfeld

Web Based Development Environments

2017-2018

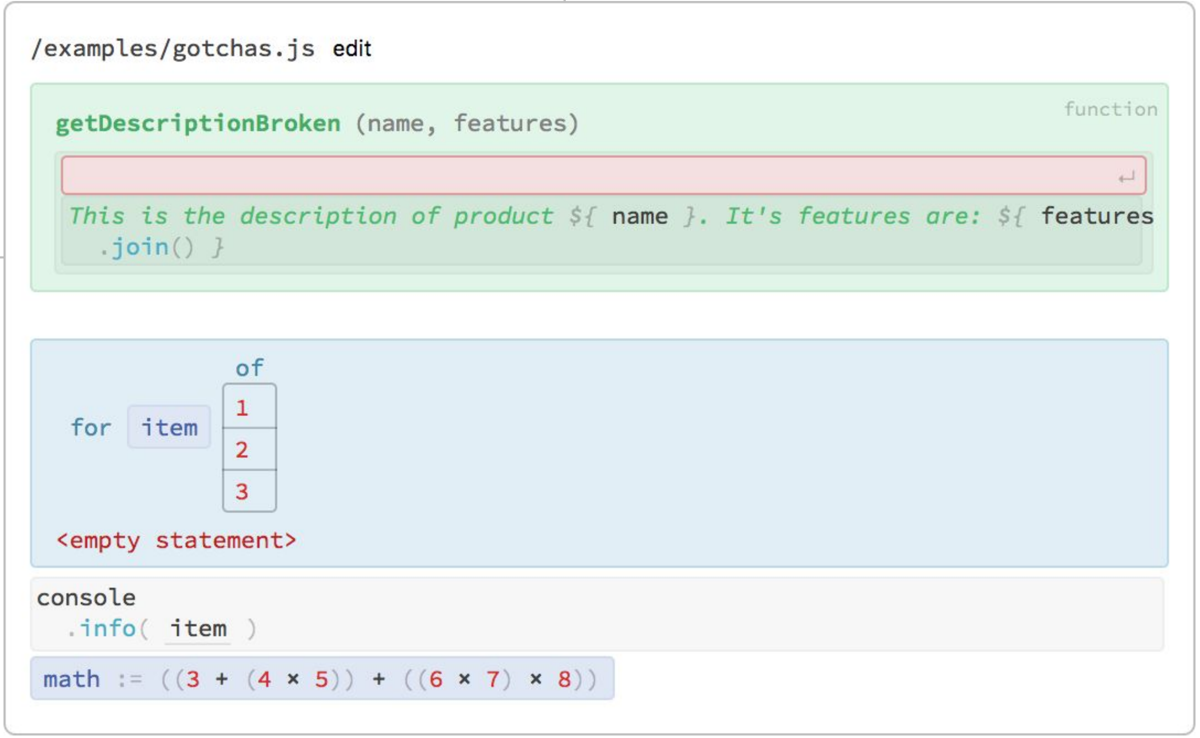
Demo

1. Add.js - Same function, several ways to format it
2. Gotchas.js - Hard to spot bugs
3. Primitives.js - Responsive layout and emojis
4. Loops.js - Custom syntax style and pretty objects
5. Gotchas.js - Fix the bugs
6. Large.js - Demonstrate good performance



/examples/gotchas.js visualize

```
function getDescriptionBroken (name, features) {
  return
    `This is the description of product ${name}.
    It's features are: ${features.join()}`
}


for (let item of [1, 2, 3]);
  console.info(item);


const math = 3 + 4 * 5 + 6 * 7 * 8

```



/examples/gotchas.js edit

```
getDescriptionBroken (name, features)                                    function

  ↵
  This is the description of product ${ name }. It's features are: ${ features
    .join() }

               of
  for  item   1
              2
              3
  <empty statement>

console
  .info( item )

math := ((3 + (4 × 5)) + ((6 × 7) × 8))
```

The Problem

1. **Plain text can be hard to comprehend**

   => Formatting & syntax highlighting try to mitigate this problem

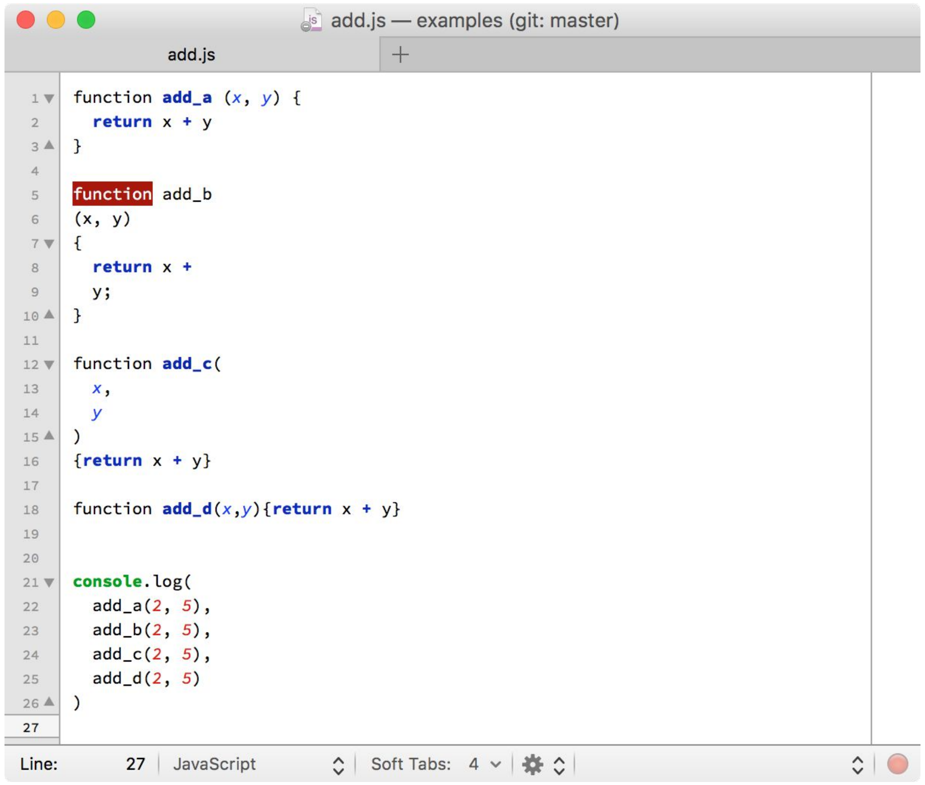
2. **Formatting is decided by the author and not the reader**

   => Code is less comprehensible in unfamiliar formatting which leads to decreased development speed and higher error rates

3. **Semantically equivalent code can be written in vastly different textual representations**

   => Makes it hard to reason about code

Instead: Syntax Visualization

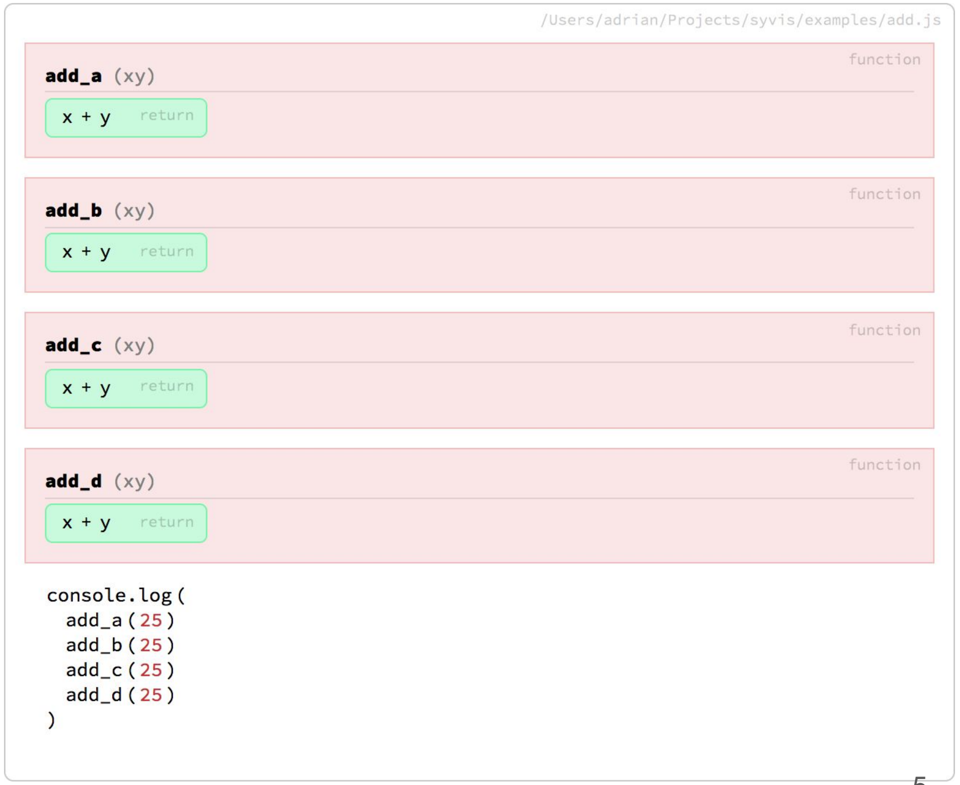


add.js — examples (git: master)

add.js

```
function add_a (x, y) {
  return x + y
}

function add_b
(x, y)
{
  return x +
  y;
}

function add_c(
  x,
  y
)
{return x + y}

function add_d(x,y){return x + y}


console.log(
  add_a(2, 5),
  add_b(2, 5),
  add_c(2, 5),
  add_d(2, 5)
)
```

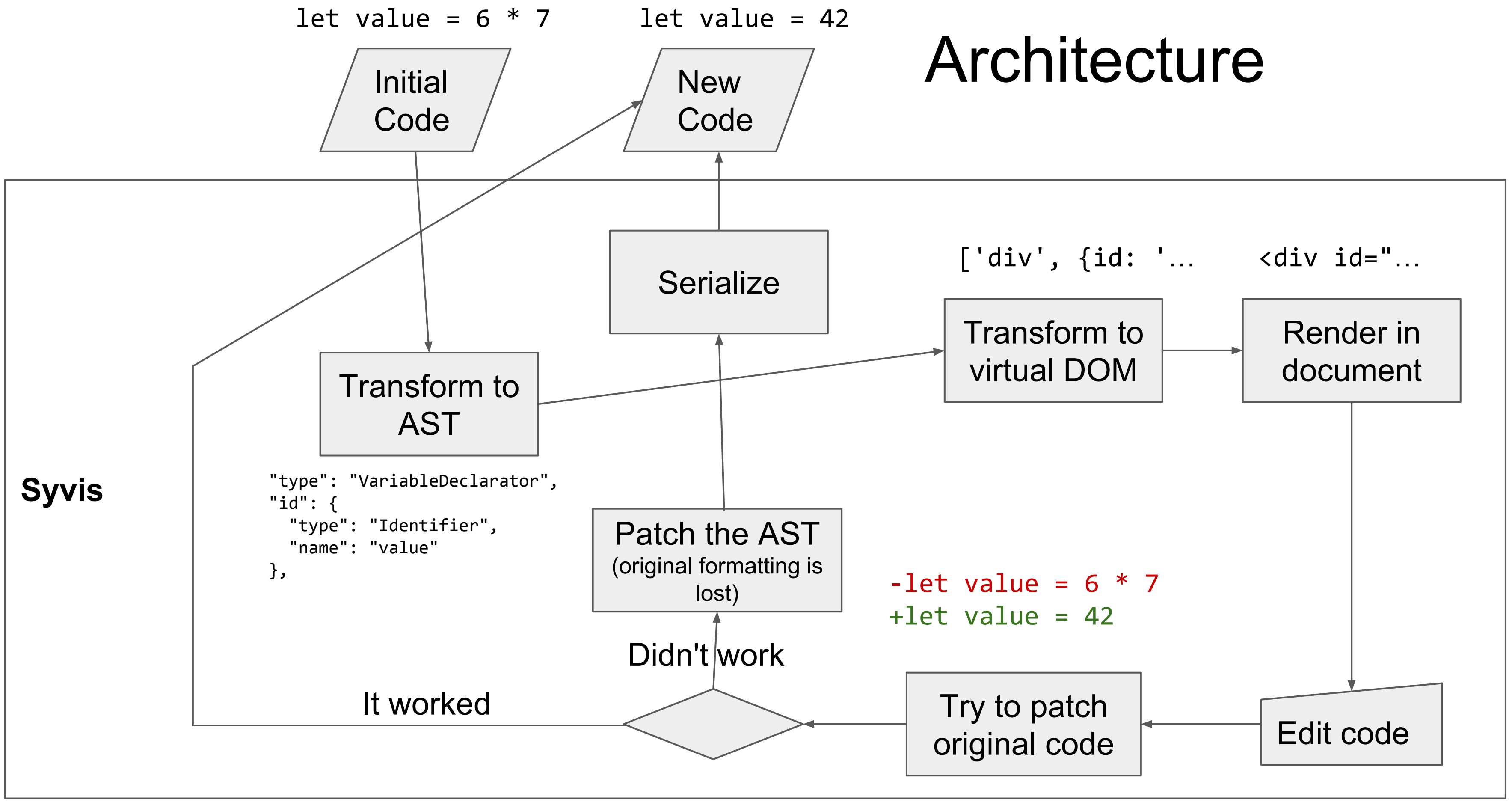
Line: 27 JavaScript Soft Tabs: 4



/Users/adrian/Projects/syvis/examples/add.js

```
add_a (xy)                function
  x + y   return

add_b (xy)                function
  x + y   return

add_c (xy)                function
  x + y   return

add_d (xy)                function
  x + y   return

console.log(
  add_a(25)
  add_b(25)
  add_c(25)
  add_d(25)
)
```

Architecture

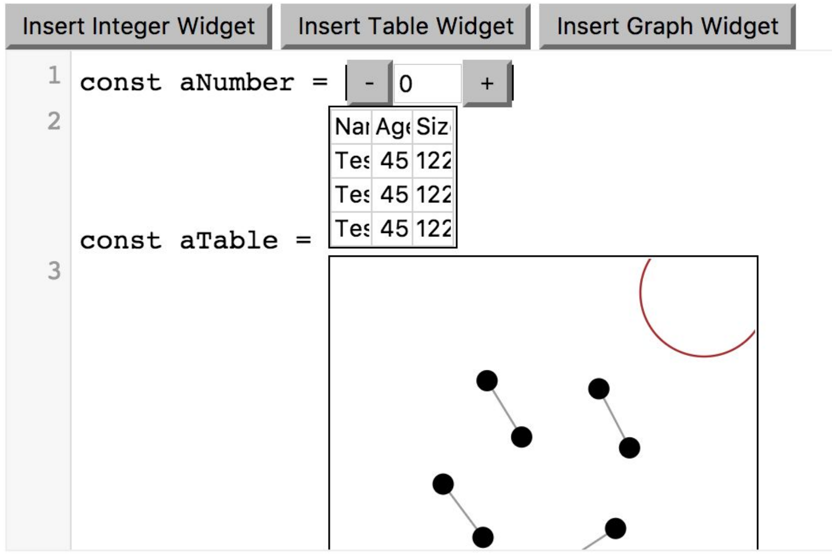


Challenges

- **Rendering is complex**
  - ~120 commits just to get it working
  - 64 node visualizers and still counting
  - (Almost) unlimited edge cases
- **Developer needs to get used to it**
- **Tooling needs to be built**

Future Work

1. **Better editing**
2. **Several Themes + Selector**
3. **Different input elements for different data types**
    a. **Slider**
    b. **Calendar widget**
4. Drag and drop reordering
5. Multi language support with integrated cross compilation
6. AST all the way
7. Multi language rendering (Math formulas, Latex, Markdown)

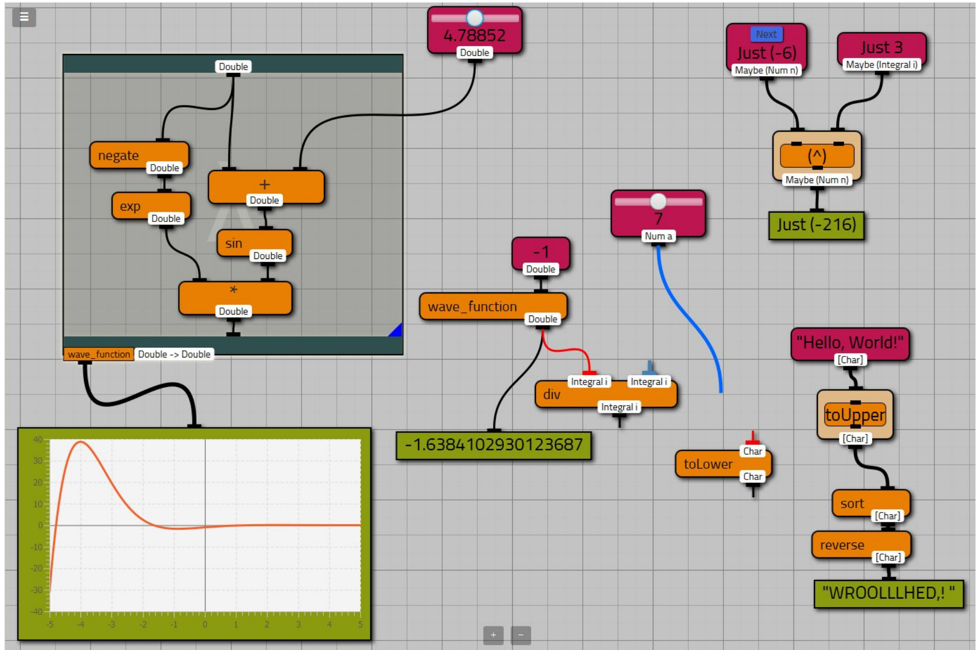


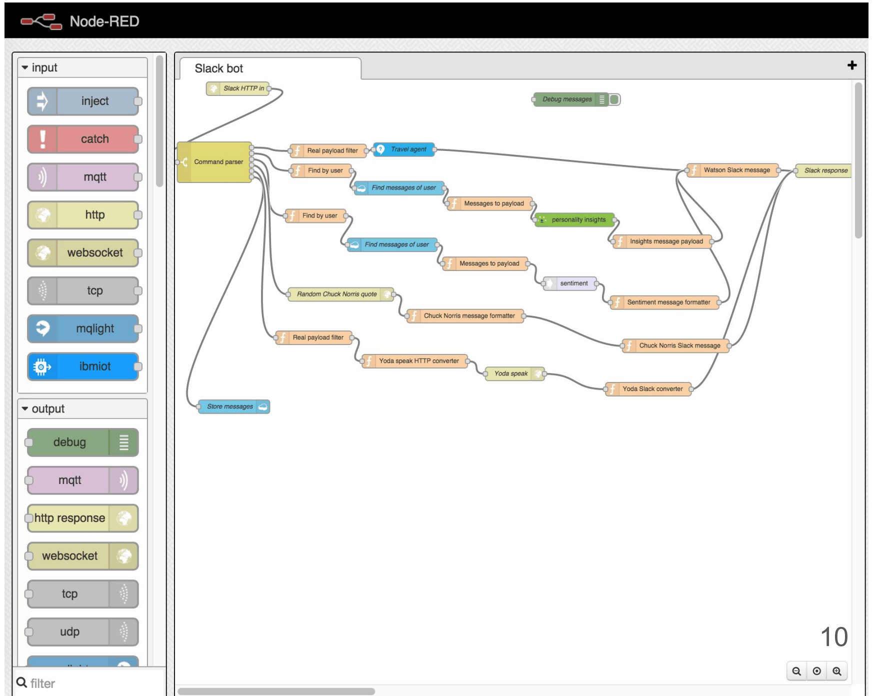
```
const aNumber = [-] 0 [+]

const aTable =
const aGraph =
```

Nar	Age	Siz
Tes	45	122
Tes	45	122
Tes	45	122

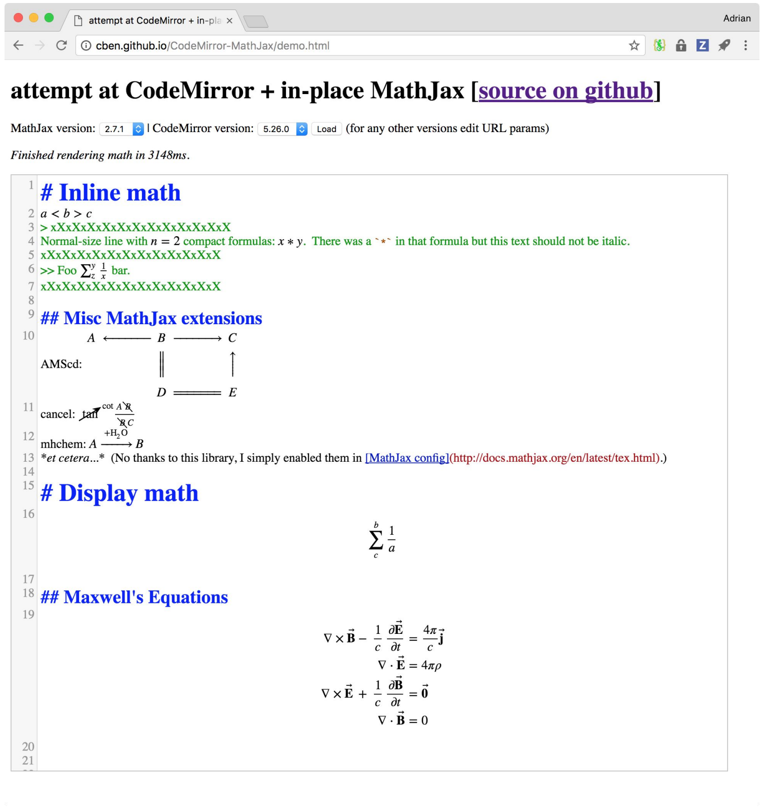
Probably Not Ideal: Flow Graphs

Still hard to read
Still unlimited different representations





Inline Widgets


attempt at CodeMirror + in-pla ×

Adrian

cben.github.io/CodeMirror-MathJax/demo.html

attempt at CodeMirror + in-place MathJax [source on github]

MathJax version: 2.7.1 | CodeMirror version: 5.26.0 Load (for any other versions edit URL params)

Finished rendering math in 3148ms.

# Inline math

$a < b > c$

> xXxXxXxXxXxXxXxXxXxXxXxX

Normal-size line with $n = 2$ compact formulas: $x * y$. There was a `` `*` `` in that formula but this text should not be italic.

xXxXxXxXxXxXxXxXxXxXxXxX

>> Foo $\sum_z^y \frac{1}{x}$ bar.

xXxXxXxXxXxXxXxXxXxXxXxX

## Misc MathJax extensions

AMScd:
$$\begin{tikzcd} A & B \arrow[l] \arrow[r] \arrow[d, equal] & C \\ & D \arrow[r, equal] & E \arrow[u] \end{tikzcd}$$

cancel: $\cancelto{\cot}{\tan} \frac{A \cancel{B}}{\cancel{B} C}$

mhchem: $A \xrightarrow{+H_2O} B$

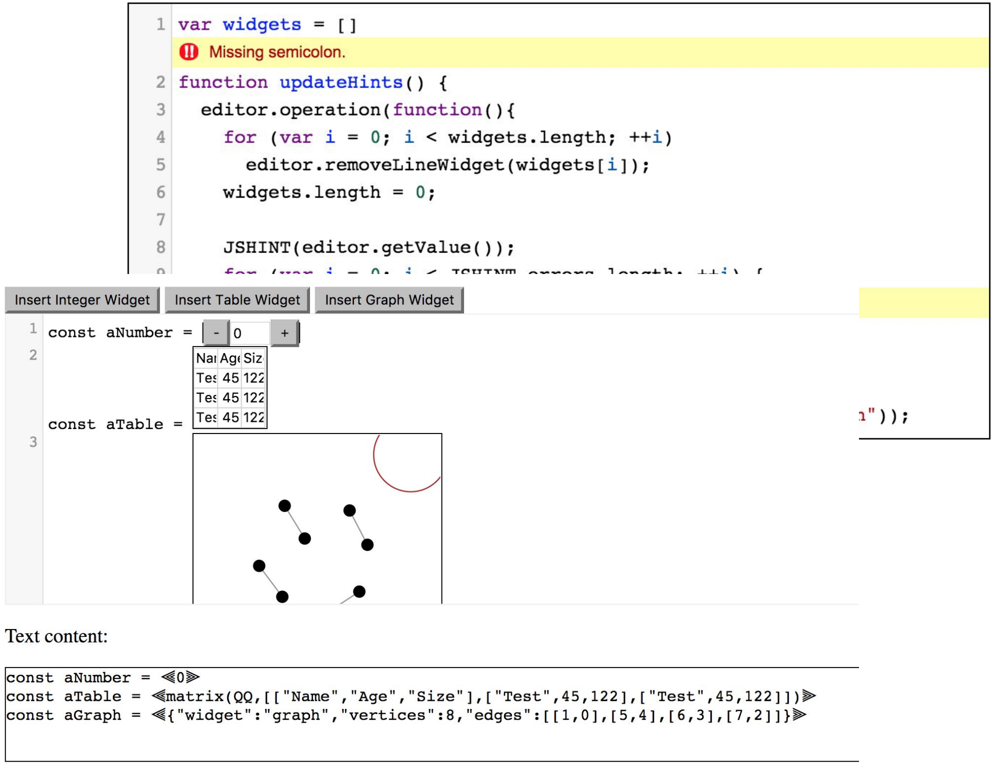
et cetera... (No thanks to this library, I simply enabled them in [MathJax config](http://docs.mathjax.org/en/latest/tex.html).)

# Display math

$$\sum_c^b \frac{1}{a}$$

## Maxwell's Equations

$$\nabla \times \vec{\mathbf{B}} - \frac{1}{c}\frac{\partial \vec{\mathbf{E}}}{\partial t} = \frac{4\pi}{c}\vec{\mathbf{j}}$$
$$\nabla \cdot \vec{\mathbf{E}} = 4\pi\rho$$
$$\nabla \times \vec{\mathbf{E}} + \frac{1}{c}\frac{\partial \vec{\mathbf{B}}}{\partial t} = \vec{\mathbf{0}}$$
$$\nabla \cdot \vec{\mathbf{B}} = 0$$


```
var widgets = []
Missing semicolon.
function updateHints() {
  editor.operation(function(){
    for (var i = 0; i < widgets.length; ++i)
      editor.removeLineWidget(widgets[i]);
    widgets.length = 0;

    JSHINT(editor.getValue());
```

```
"));
```

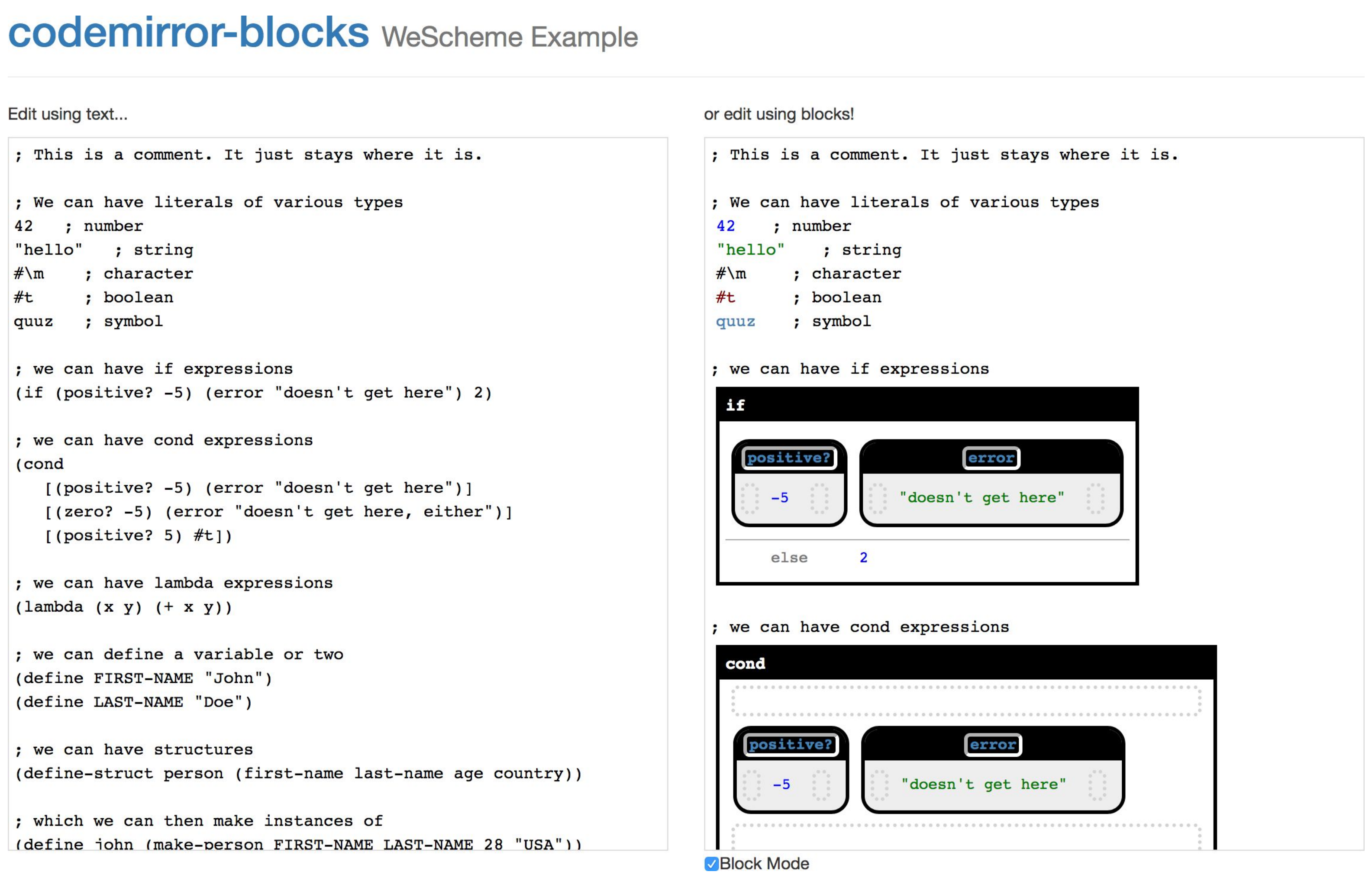
Insert Integer Widget | Insert Table Widget | Insert Graph Widget

```
const aNumber = [ - | 0 | + ]
const aTable =
const aGraph =
```

Nai	Agı	Siz
Tes	45	122
Tes	45	122
Tes	45	122

Text content:

```
const aNumber = ⋘0⋙
const aTable = ⋘matrix(QQ,[["Name","Age","Size"],["Test",45,122],["Test",45,122]])⋙
const aGraph = ⋘{"widget":"graph","vertices":8,"edges":[[1,0],[5,4],[6,3],[7,2]]}⋙
```



codemirror-blocks WeScheme Example

Edit using text...

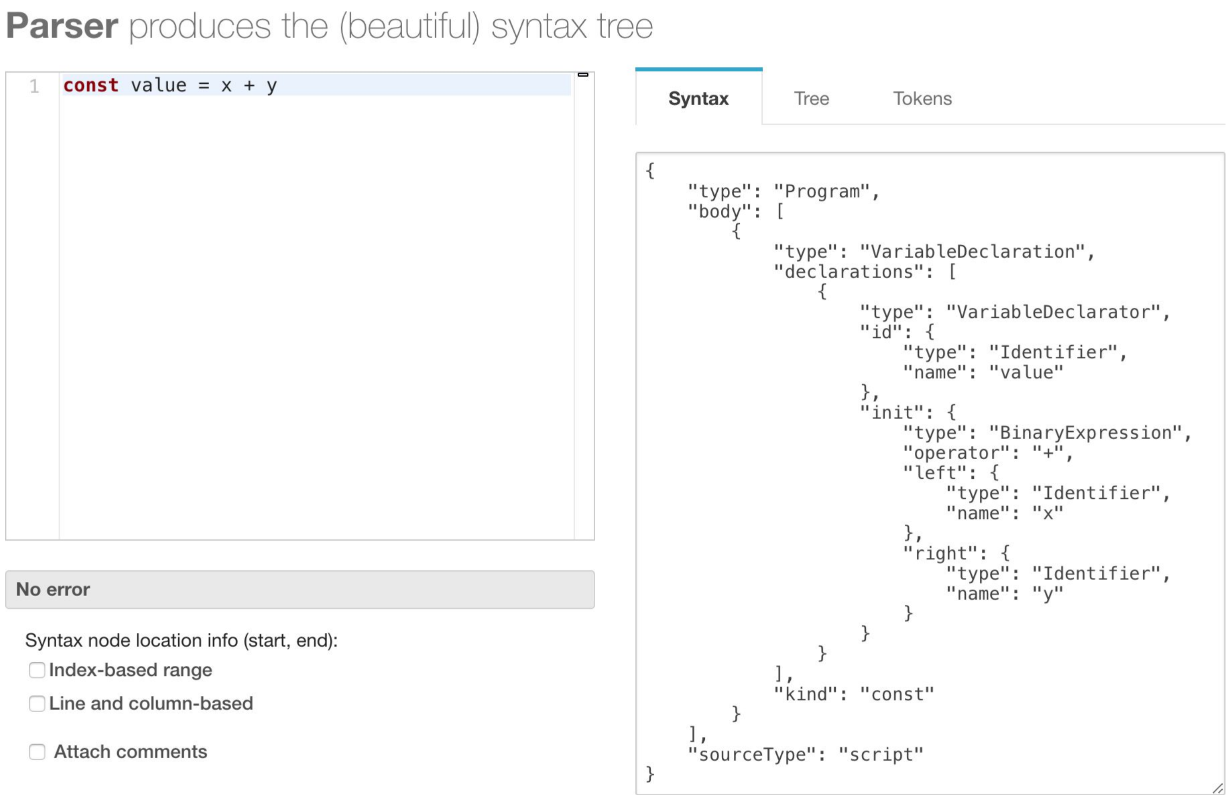
```
; This is a comment. It just stays where it is.

; We can have literals of various types
42    ; number
"hello"    ; string
#\m     ; character
#t      ; boolean
quuz    ; symbol

; we can have if expressions
(if (positive? -5) (error "doesn't get here") 2)

; we can have cond expressions
(cond
   [(positive? -5) (error "doesn't get here")]
   [(zero? -5) (error "doesn't get here, either")]
   [(positive? 5) #t])

; we can have lambda expressions
(lambda (x y) (+ x y))

; we can define a variable or two
(define FIRST-NAME "John")
(define LAST-NAME "Doe")

; we can have structures
(define-struct person (first-name last-name age country))

; which we can then make instances of
(define john (make-person FIRST-NAME LAST-NAME 28 "USA"))
```

or edit using blocks!

```
; This is a comment. It just stays where it is.

; We can have literals of various types
42    ; number
"hello"    ; string
#\m     ; character
#t      ; boolean
quuz    ; symbol

; we can have if expressions
if
  positive?  -5
  error  "doesn't get here"
  else  2

; we can have cond expressions
cond
  positive?  -5
  error  "doesn't get here"
```

☑ Block Mode

Use AST (not the Code)



Parser produces the (beautiful) syntax tree

```
const value = x + y
```

No error

Syntax node location info (start, end):

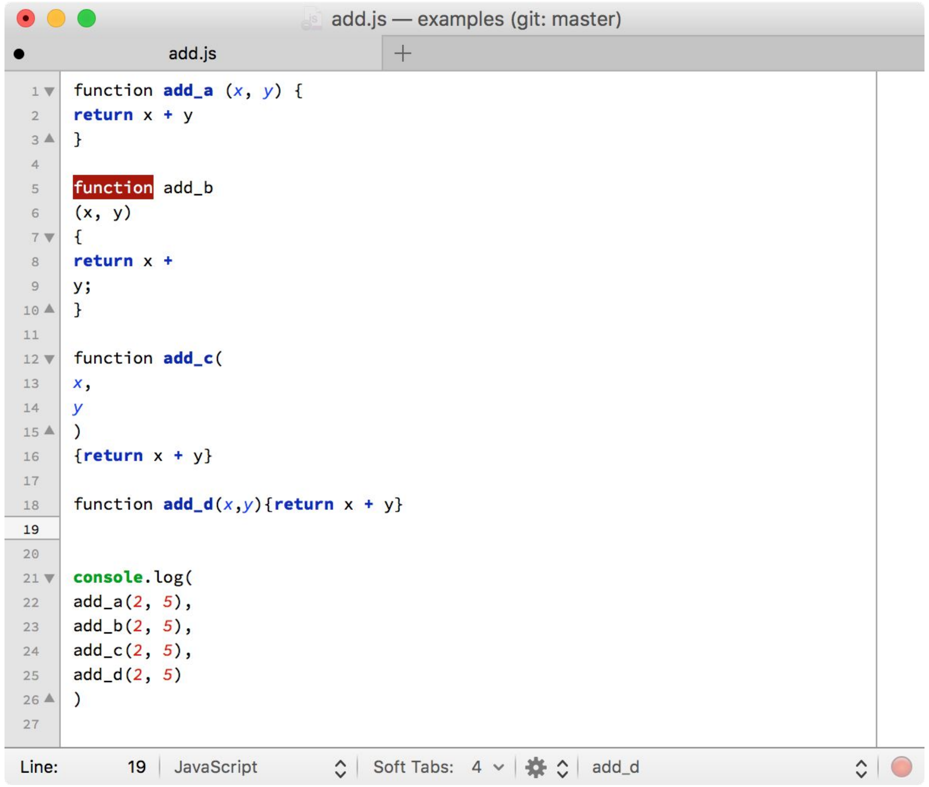
☐ Index-based range

☐ Line and column-based

☐ Attach comments

```
{
    "type": "Program",
    "body": [
        {
            "type": "VariableDeclaration",
            "declarations": [
                {
                    "type": "VariableDeclarator",
                    "id": {
                        "type": "Identifier",
                        "name": "value"
                    },
                    "init": {
                        "type": "BinaryExpression",
                        "operator": "+",
                        "left": {
                            "type": "Identifier",
                            "name": "x"
                        },
                        "right": {
                            "type": "Identifier",
                            "name": "y"
                        }
                    }
                }
            ],
            "kind": "const"
        }
    ],
    "sourceType": "script"
}
```

Formatting Should be a User Setting

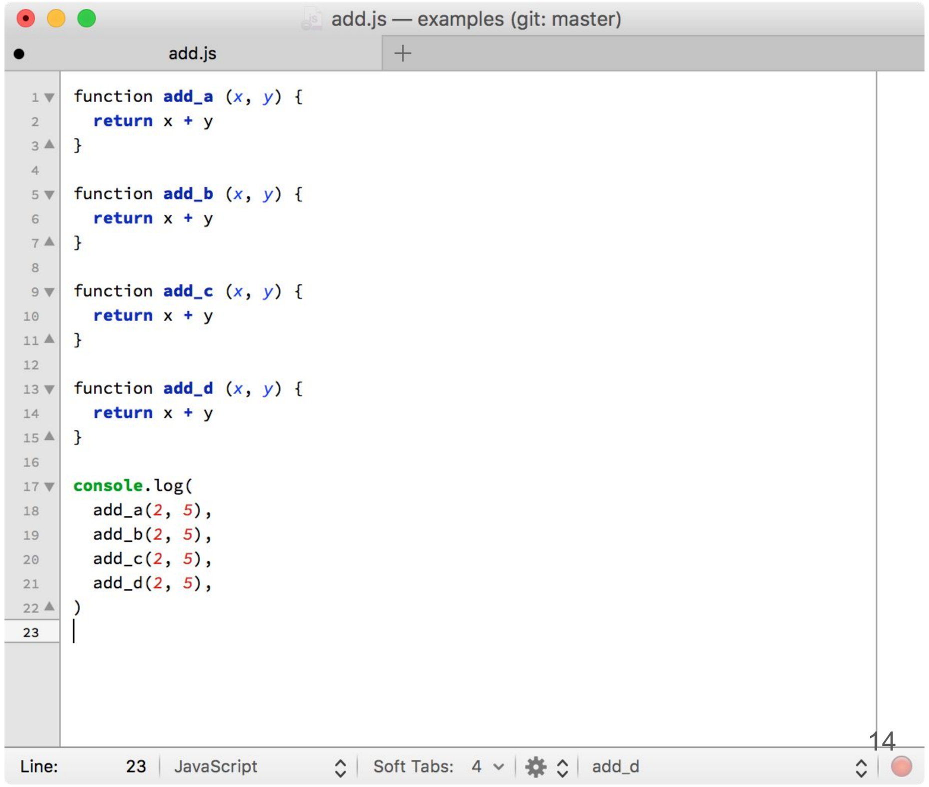


add.js — examples (git: master)

add.js

```
function add_a (x, y) {
return x + y
}

function add_b
(x, y)
{
return x +
y;
}

function add_c(
x,
y
)
{return x + y}

function add_d(x,y){return x + y}


console.log(
add_a(2, 5),
add_b(2, 5),
add_c(2, 5),
add_d(2, 5)
)

```

Line: 19 JavaScript Soft Tabs: 4 add_d



add.js — examples (git: master)

add.js

```
function add_a (x, y) {
  return x + y
}

function add_b (x, y) {
  return x + y
}

function add_c (x, y) {
  return x + y
}

function add_d (x, y) {
  return x + y
}

console.log(
  add_a(2, 5),
  add_b(2, 5),
  add_c(2, 5),
  add_d(2, 5),
)

```

Line: 23 JavaScript Soft Tabs: 4 add_d