

Introducing Repla: A Live-Coding Tool

How to make innovating programming tools that are compatible with existing workflows.

Roben Kleene

[@robenkleene](https://twitter.com/robenkleene)

robenkleene.com

github.com/robenkleene/repla-live-coding

Who Am I

- >> Developer for Apple platforms (iOS & Mac)
- >> Currently making a programming tool called Repla
- >> Left job managing WSJ iOS app team to make it
- >> Not a researcher, trying to create a business, not a boundary pusher

Simple Poll APP 4:46 AM

The thing I envision for the future of coding is for:

1 Programmers 16

@Miles Sabin, @David Piepgrass, @yairchu, @Wouter, @robenkleene, @Will,
@Kartik Agaram, @John Austin, @Brian Hempel, @missingfaktor, @Tony Cheal,
@wtaysom, @Stathis, @Vladimir Gordeev, @gman, @piggy1

2 Non Programmers 15

@Philipp Krüger, @Stefan, @ahmedr, @Niluka Satharasinghe, @yairchu,
@Duncan Cragg, @Jacob Sandlund, @Peter Abrahamsen, @Scott Anderson,
@jonathoda, @missingfaktor, @Tony Cheal, @Robin, @Nick Smith, @Tim Swast

1

2

Delete Poll

Simple Poll

Edit Settings



+2

19 replies Last reply 26 days ago

Traits of Successful Programming Tools (For Existing Programmers)

1/2

- » **Browser UI:** A lingua franca for graphics.
- » **Language Agnostic:** Work with existing programming languages.
- » **Plain Text:** An open data format.
- » **Packages Written in Scripting Languages:** Make customizations easy to share and modify.

Traits of Successful Programming Tools (For Existing Programmers)

2/2

- » **Text Editors:** The concentration of programming man-hours.
- » **Unix Processes:** Use child processes to extend capabilities.
- » **Version Control:** State of the art collaboration.

What do users have to give up to use your tool?

- » Text editors?
- » Version control?

Is your solution more useful than these things?

Case Study: Visual Studio Code

Visual Studio Code is a **language agnostic text editor** used to edit **plain text** files that can be stored in **version control**. It's user interface is displayed using a **browser rendering engine**. Its packages are written in scripting languages that run **Unix processes**.

Visual Studio Code Market Share

>> 2016: 7.2%

>> 2017: 22.3%

>> 2018: 34.9%

>> 2019: 50.7%

Stack Overflow Insights.

Competes with text editors: 20–25 employees.

Why Not Integrate
With an IDE?



Console Debug Main.storyboard

Repla

Repla

Main.storyboard

Application Scene

Window Controller Scene

Split Web View Controller Scene

Split Web View Controller

- Split View
- Split View Item
- Split View Item
- First Responder
- Relationship "split items" to "Web View Controller"
- Relationship "split items" to "Web View Controller"

Web View Controller Scene

Web View Controller Scene

Add New Alignment Constraints

- Leading Edges
- Trailing Edges
- Top Edges
- Bottom Edges
- Horizontal Centers
- Vertical Centers
- First Baselines
- Horizontally in Container
- Vertically in Container

View as: Dark Appearance

Auto Filter

All Output Filter

Storyboard

My Mac

Repla | Build Repla: Succeeded | Yesterday at 3:08 PM

Identity and Type

Name Main.storyboard

Type Default - Interface Builder...

Location Relative to Group

Main.storyboard

Full Path /Users/robenkleene/Development/Projects/Cocoa/Repla/Repla/Main.storyboard

On Demand Resource Tags

Tags

Document

Latest Xcode (10.0)

Deployment Target (10.14)

Use Auto Layout

Localization

Relationships

Constraints

When to integrate with an IDE?

If your feature fits into existing user interface features and requires no interaction.

- >>  Linters
- >>  Splits & Folds

Repla

What is Repla?

- >> A web rendering engine
- >> Unix process management
- >> A packaging system
- >> Not an editor, works alongside existing tools
- >> A platform

Well maybe try to replace one thing... the browser

- » Sounds crazy?
- » The developer trio: a **browser**, a **text editor**, and a **terminal**.
- » A text editor and a terminal both use a **packaging system** to extend functionality by running **Unix processes**. The browser?

Screenshots

A screenshot of a Mac OS X terminal window titled "example.js — Files (git: master)". The window has a dark grey header bar with red, yellow, and green close buttons. The main area displays the following code:

```
function addNumbers(x, y) {
    return x + y;
}

addNumbers(1, 2);
```

The code editor interface includes a vertical line number column on the left, a status bar at the bottom with "Line: 1 | JavaScript", and a toolbar with "Soft Tabs: 2" and "Symbols" buttons.

A screenshot of a Mac OS X terminal window titled "example.rb — Files (git: master)". The window has a dark grey header bar with red, yellow, and green close buttons. The main area displays the following code:

```
def add_numbers(x, y)
    return x + y
end

add_numbers(1, 2)
```

The code editor interface includes a vertical line number column on the left, a status bar at the bottom with "Line: 1 | Ruby", and a toolbar with "Soft Tabs: 2" and "Symbols" buttons. A tooltip "add_numbers..." is visible near the bottom right of the status bar.

The screenshot shows a Mac OS X application window titled "example.html — Files (git: master)". The window has a standard OS X title bar with red, yellow, and green close, minimize, and zoom buttons. The main content area displays the following HTML code:

```
<section>
    <p>The HTML displays a live preview of an HTML
document.</p>
    <!-- <p>If you edit the document and save, the preview
refreshes.</p> -->
</section>
```

The code editor interface includes a vertical scroll bar on the left, line numbers 21 through 26 at the top, and a status bar at the bottom showing "Line: 23 | HTML" and "Tab Size: 4".

The screenshot shows a Mac OS X application window titled "example.md — Files (git: master)". The window has a standard OS X title bar with red, yellow, and green close, minimize, and zoom buttons. The main content area displays the following Markdown code:

```
**Markdown Plugin Demo**

The Markdown plugin displays a live preview of a Markdown
document render to HTML.

<!-- If you edit the document and save, the preview refreshes. -->
```

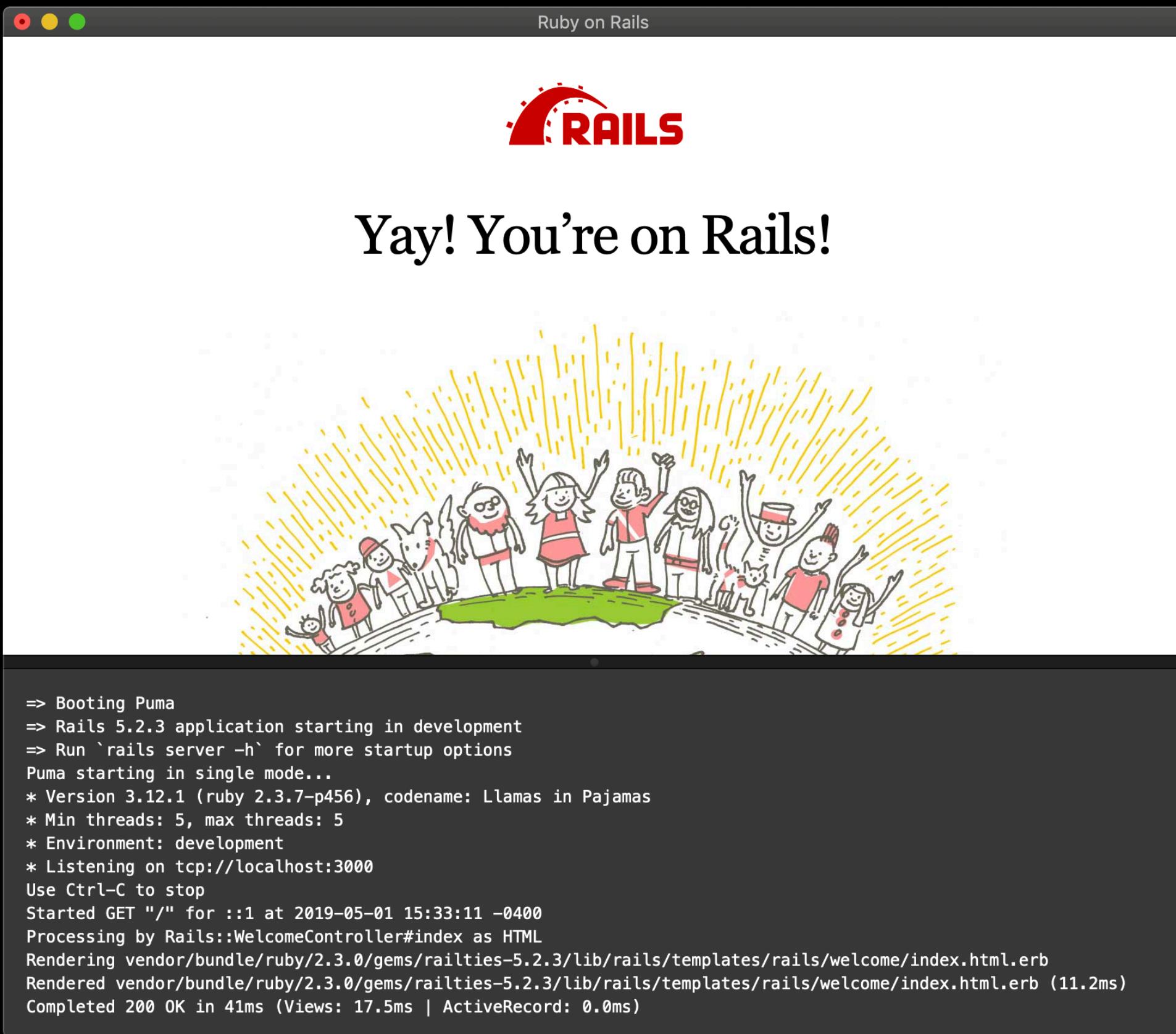
The code editor interface includes a vertical scroll bar on the left, line numbers 1 through 5 at the top, and a status bar at the bottom showing "Line: 1 | Markdown" and "Tab Size: 4".

Your Text Editor

```
1 function addNumbers(a, b) {  
2     return a + b;  
3 }  
4  
5 var left = 1;  
6  
7 var right = 2;  
8  
9 addNumbers(left, right);  
10
```

Repla

```
function addNumbers(a, b) {  
    return a + b;  
}  
=> undefined  
var left = 1;  
=> undefined  
var right = 2;  
=> undefined  
addNumbers(left, right);  
=> 3
```



Roadmap

1. **Browser & Processes:** Web Development (integrate the server and browser into one window, automatically refresh)
2. **Packaging System:** Distributing Web Apps (one click install, one click run, e.g., Jupyter Notebooks)
3. **Packages 1:** Live Coding View
4. **Packages 2:** Framer Classic & Processing

Use Cases

The image shows three separate Mac OS X windows side-by-side, all titled "samplemarkdown.md".

- Left Window:** Displays the rendered content of the Markdown file. It contains:
 - A large, bold, dark gray header: "Marked".
 - A section titled "What's Markdown?".
 - A detailed paragraph explaining Marked's compatibility with various processors and its included resources (Markdown Basics, TUAW Markdown Primer, MultiMarkdown User's Guide).
 - A note about the preview application being editor-agnostic and watching files for changes.
- Middle Window:** Shows the raw Markdown source code. It includes line numbers (1 through 13) and the following text:

```
1 #·Marked
2
3 ##·What's·Markdown?
4
5 Marked·works·with·Markdown·and·MultiMarkdown·,·and·can·convert·syntax
extended·to·work·with·just·about·any·processor·you·need·,·includin
assume·that--since·you're·here--you·at·least·know·what·these·mark
Gruber's·[Markdown·Basics]·[daringfireball]·,·check·out·the·[TUAW·M
with·the·[MultiMarkdown·User's·Guide]·[github]·.·Marked·includes·a·
you·can·brush·up·as·you·go.
6
7 [daringfireball]:·http://daringfireball.net/projects/markdown/bas
8 [github]:·http://fletcher.github.com/peg-multimarkdown/mmd-manua
9 [tuaw]:·http://www.tuaw.com/markdown-primer
10 [help]:·http://markedapp.com/help/
11
12 ##·What's·Marked?
13
```
- Right Window:** Similar to the middle window, it also displays the raw Markdown source code with line numbers 1 through 13.

Marked

- >>  Plain text
- >>  Works with text editors
- >>  Works with version control
- >>  Not language agnostic

See also: Deckset

The screenshot shows a Mac OS X style window titled "Soulver". The main content area contains a table comparing interest rates for two banks. The table has two columns: descriptions on the left and amounts on the right.

Comparing Interest Rates:	
Bank 1: \$50,000 + 5.25%	\$52,625.00
Bank 2: \$50,000 + 6.25%	\$53,125.00
Difference of	
\$53,125.00 - \$52,625.00	\$500.00
Total: \$106,250.00	

Soulver

- >> 🤢 Not plain text
- >> 🤢 Not language agnostic
- >> 🤢 Doesn't work with text editors
- >> 🤢 Doesn't work with version control

See also: Calca

The screenshot shows a web browser window for observablehq.com. The title bar includes standard OS X icons and the URL 'observablehq.com'. The main header features the 'Observable' logo, a search bar, and buttons for 'Demo', 'Fork', and 'Sign in'. A yellow banner at the top says 'Welcome. This is live code! Click the left margin to view or edit.' with a close button 'X'. Below this, the notebook title 'Five-Minute Introduction' is displayed, along with its author 'Mike Bostock' and creation date 'Dec 4, 2017'. It has 35 forks and 90 likes. The notebook content starts with a text cell: 'Welcome to Observable! This notebook gives a quick but technical overview of Observable's features. For a slightly longer introduction, see [Introduction to Notebooks](#) and [introductory tutorial series](#). If you want to watch rather than read, we've also posted a [short video introduction](#)!' followed by a code cell showing the calculation $2 * 3 * 7$ resulting in 42. The next cell contains the following JavaScript code:

```
5050
{
  let sum = 0;
  for (let i = 0; i <= 100; ++i) {
    sum += i;
  }
  return sum;
}
```

Observable

- >> 🤢 Not plain text
- >> 🤢 Not language agnostic
- >> 🤢 Doesn't work with text editors
- >> 🤢 Doesn't work with version control

See also: Jupyter Notebooks, Swift Playgrounds, R Markdown

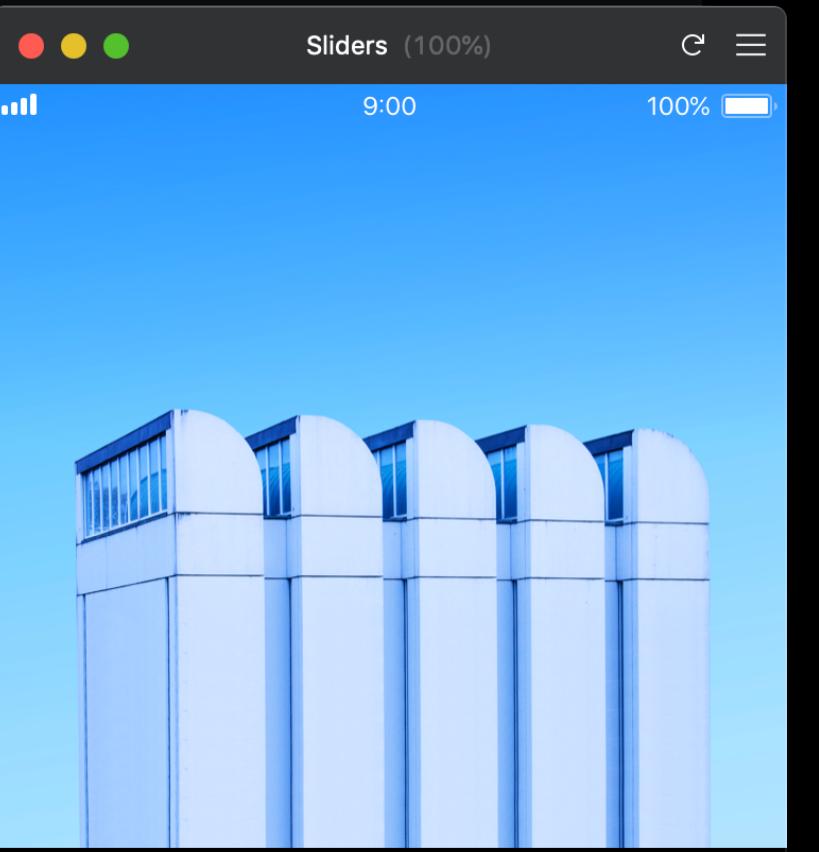
Slider.tsx

```
import * as React from "react"
import { Frame, PropertyControls, ControlType } from "framer"

interface Props {
  value: number
  knob: string
  track: string
  tint: string
  width: number
  height: number
}

export class Slider extends React.Component<Props> {
  static defaultProps = {
    width: 120,
    height: 44,
    value: 50,
    knob: "#FFF",
    track: "#333",
    tint: "#8DF",
  }

  static propertyControls: PropertyControls = {
    value: { type: ControlType.Slider },
    tint: { type: ControlType.Color },
  }
}
```



Framer Classic

- >>  Not plain text
- >>  Not language agnostic
- >>  Doesn't work with text editors
- >>  Doesn't work with version control

See also: Processing

Live Coding With Repla

Advantages

- >> Language packages: IRB.replaplugin, Python.replaplugin, Node.replaplugin
- >> Use your existing code editor, with regular file extensions: .rb, .py, .js
- >> Check them into version control

Live Coding Implementation

```
def initialize(command)
  PTY.spawn(command) do |output, input, _pid|
    Thread.new do
      output.each do |line|
        output_controller.parse_output(line)
      end
    end
    @input = input
  end
end
```

```
def parse_input(input)
  input_controller.parse_input(input)
  @input.write(input)
end
```

Other Implementation Details

» The Ruby process watches the file system (`gem 'listen'`)

Introducing Repla: A Live-Coding Tool

How to make innovating programming tools that are compatible with existing workflows.

That's it, thanks!

Roben Kleene

@robenkleene

robenkleene.com

github.com/robenkleene/repla-live-coding