

# Introducing Repla: A Live-Coding Tool

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

Roben Kleene

@robenkleene

robenkleene.com

[github.com/robenkleene/repla-live-coding](https://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
- I like AppKit GUI apps, and Unix TUI apps
- 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**. Its 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

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

Localize...

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 code editor window titled "example.js — Files (git: master)". The window has a dark theme with light-colored code. The code is as follows:

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

addNumbers(1, 2);
```

The editor interface includes a vertical line number column on the left, a toolbar at the top with red, yellow, and green buttons, and a status bar at the bottom with the line number "Line: 1", the language "JavaScript", and settings for "Soft Tabs: 2" and "Symbols".

A screenshot of a code editor window titled "example.rb — Files (git: master)". The window has a dark theme with light-colored code. The code is as follows:

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

add_numbers(1, 2)
```

The editor interface includes a vertical line number column on the left, a toolbar at the top with red, yellow, and green buttons, and a status bar at the bottom with the line number "Line: 1", the language "Ruby", and settings for "Soft Tabs: 2" and "add\_numbers(...)".

The screenshot shows a Mac OS X application window titled "example.html — Files (git: master)". The window has a dark grey header bar with standard OS X window controls (red, yellow, green) and a title bar. The main content area displays an HTML document with syntax highlighting. The code is as follows:

```
<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 dark grey header bar with standard OS X window controls and a title bar. The main content area displays a Markdown document with syntax highlighting. The code is as follows:

```
**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. -->
```

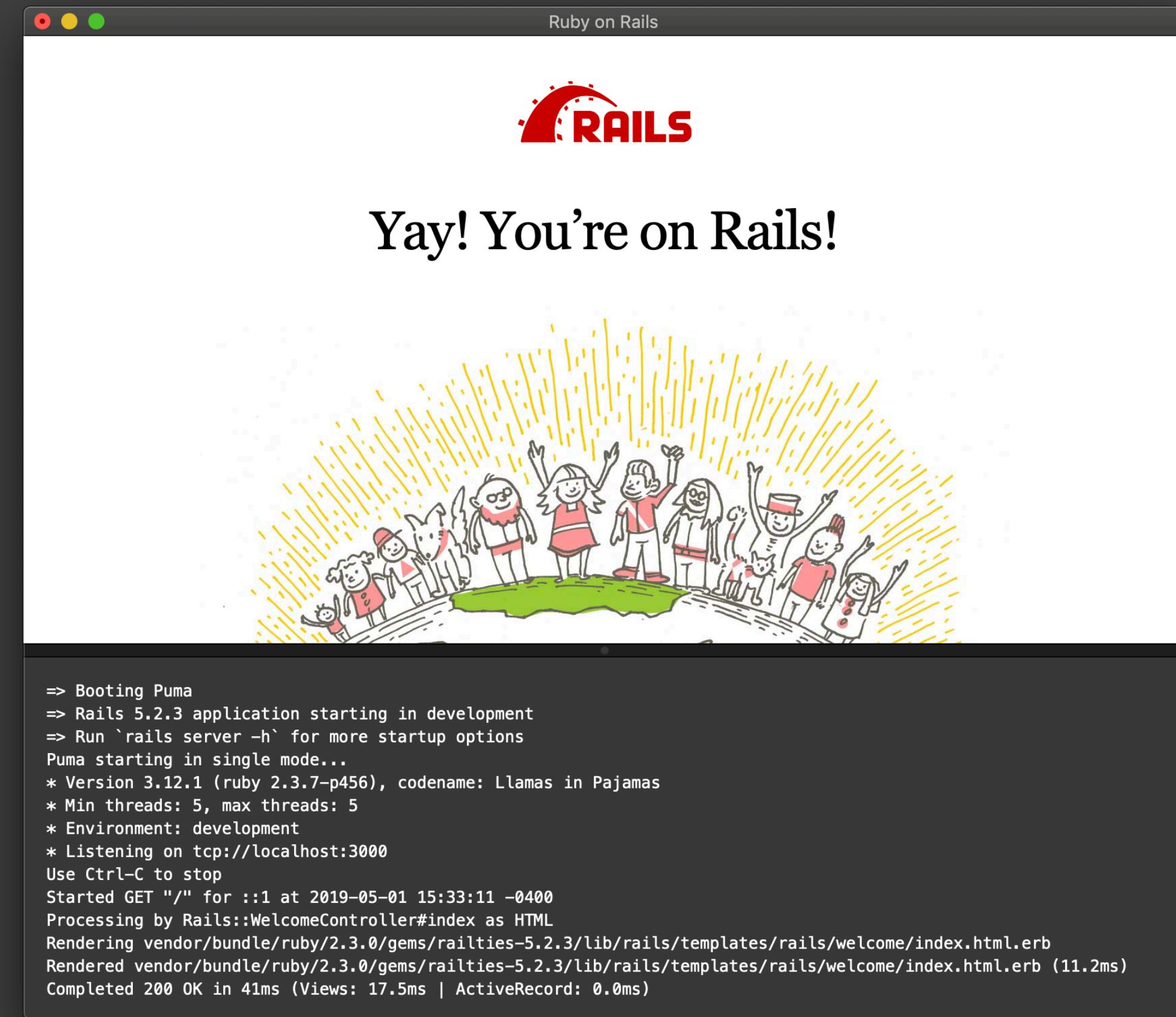
The code editor interface includes a vertical scroll bar on the left, line numbers 1 through 5 at the top, and a status bar at the bottom showing "Line: 1 | Markdown" and "Tab Size: 4".

## 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 that Marked supports Markdown and MultiMarkdown, and can convert syntax from several variants. It mentions Textile, reStructuredText, Wikitext, and more. It suggests starting with John Gruber's [Markdown Basics](#) and the [TUAW Markdown Primer](#), and brushing up on extended syntax with the [MultiMarkdown User's Guide](#). It notes that a full Markdown syntax guide is available under the help menu.
- Middle Window:** Shows the raw Markdown source code. Lines 1 through 13 are visible:

```
1 #· Marked
2
3 ##· What's· Markdown?
4
5 Marked· works· with· Markdown· and· MultiMarkdown· ,· and· can· convert· syn
       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-manual
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, showing the raw Markdown source code for the first few lines.

# Marked

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

See also: Deckset

The screenshot shows a window titled "Soulver" with a dark gray header bar featuring three colored circular icons (red, orange, green) on the left and a close button on the right. The main content area has a light gray background and contains the following text:

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 page title is "Observable". The main content is the "Five-Minute Introduction" notebook. At the top, there's a welcome message: "Welcome. This is live code! Click the left margin to view or edit." Below that, the notebook details are shown: "Observable · Dec 4, 2017 · The magic notebook for visualization. By Mike Bostock". There are 35 forks and 90 likes. The notebook title is "Five-Minute Introduction". The introduction text reads: "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](#)!" The next cell shows the result of the expression "2 \* 3 \* 7" which is "42". The final 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 },
  }
}
```

Sliders (100%)

9:00

100%

Saturate

Contrast

Grayscale

# 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](https://github.com/robenkleene/repla-live-coding)