Rethinking Online Code Editors for Supporting Time-based Web Documents

Rodrigo Laiola Guimarães (IBM Research | Brazil)
What are online code editors?
What are online code editors?

• Authoring tools that allow developers to code and see the results promptly
What are online code editors?

• Authoring tools that allow developers to code and see the results promptly

What are online code editors?

• Authoring tools that allow developers to code and see the results promptly


• There are a number of Web playgrounds for prototyping HTML, CSS and JavaScript
In this work…
In this work...

• We go a step further by giving special attention to the **temporal aspect** (e.g. preserving presentation state between code changes)
In this work...

• We go a step further by giving special attention to the **temporal aspect** (e.g. preserving presentation state between code changes)

• Our proof of concept has been developed using many open source libraries, and currently it works in modern Web browsers (e.g., Safari, Firefox and Chrome)
In this work...

- We go a step further by giving special attention to the **temporal aspect** (e.g. preserving presentation state between code changes)

- Our proof of concept has been developed using many open source libraries, and currently it works in modern Web browsers (e.g., Safari, Firefox and Chrome)

- Main functionalities: immediate feedback, coding assistance, playback control and programmatic visualization
Demo Video
User Interface
User Interface

```html
1. <!-- So we will have an animated background with 5 clouds moving across the screen. -->
2. Steps:
3. 1. create the clouds
4. 2. Animate them to move across the screen
5. 3. Stylize the clouds (can be done as step #2 also)
6. -->
7. <div id="clouds">
8.   <div class="cloud x1"></div>
9.   <!-- Time for multiple clouds to dance around -->
10.  <div class="cloud x2"></div>
11.  <div class="cloud x3"></div>
12.  <div class="cloud x4"></div>
13.  <div class="cloud x5"></div>
14. </div>
15. <!-- That's the markup! -->
16. <!-- That looks cool. We are done! -->
17. <!-- Disregard -->
18. <a href="https://plus.google.com/111052275277622928148?rel=me">Google+</a>
```
User Interface

---

```html
<!--
So we will have an animated background with 5 clouds moving across the screen.
Steps:
1. Create the clouds
2. Animate them to move across the screen
3. Stylize the clouds (can be done as step #2 also)
-->
<div id="clouds">
  <div class="cloud x1"></div>
  <!-- Time for multiple clouds to dance around -->
  <div class="cloud x2"></div>
  <div class="cloud x3"></div>
  <div class="cloud x4"></div>
  <div class="cloud x5"></div>
</div>
<!-- That's the markup! -->
<!-- That looks cool. We are done! -->
<!-- Disregard -->
<a href="https://plus.google.com/111052275277622928148?rel=me">Google+</a>
```

A

**CodeMirror**: text editor in JS
User Interface

<!-- So we will have an animated background with 5 clouds moving across the screen. -->

1. Create the clouds
2. Animate them to move across the screen
3. Stylize the clouds (can be done as step #2 also)

```html
<div id="clouds">
  <div class="cloud x1"></div>
  <!-- Time for multiple clouds to dance around -->
  <div class="cloud x2"></div>
  <div class="cloud x3"></div>
  <div class="cloud x4"></div>
  <div class="cloud x5"></div>
</div>

<!-- That's the markup! -->

<!-- That looks cool. We are done!! -->

<!-- Disregard -->
<a style="position: fixed; bottom: 10px; right: 10px; color: #CCC;" href="https://plus.google.com/111052275277622928148?rel=me">Google+</a>
```

**A**

CodeMirror: text editor in JS

**B**

IFRAME
User Interface

A

CodeMirror: text editor in JS

B

IFRAME
User Interface

```
1. So we will have an animated background with 5 clouds moving across the screen.
2. Steps:
   1. Create the clouds
   2. Animate them to move across the screen
   3. Stylize the clouds (can be done as step #2 also)

---

```<div id="clouds">
  <div class="cloud x1"></div>
  <!-- Time for multiple clouds to dance around -->
  <div class="cloud x2"></div>
  <div class="cloud x3"></div>
  <div class="cloud x4"></div>
  <div class="cloud x5"></div>
</div>""/>

```

A

CodeMirror: text editor in JS

B

IFRAME
User Interface

A

CodeMirror: text editor in JS

B

IFRAME
User Interface

1. HTML
2. CSS
3. JS
4. Timesheet

So we will have an animated background with 5 clouds moving across the screen.

Steps:
1. Create the clouds
2. Animate them to move across the screen
3. Stylize the clouds (can be done as step #2 also)

```html
<div id="clouds">
  <div class="cloud x1"></div>
  <!-- Time for multiple clouds to dance around -->
  <div class="cloud x2"></div>
  <div class="cloud x3"></div>
  <div class="cloud x4"></div>
  <div class="cloud x5"></div>
</div>

<!-- That's the markup! -->
<!-- That looks cool. We are done! -->
<!-- Disregard -->
```

A: CodeMirror: text editor in JS
B: IFRAME
Infrastructure

```html
1. So we will have an animated background with 5 clouds moving across the screen.
2. Steps:
   1. Create the clouds
   2. Animate them to move across the screen
   3. Stylize the clouds (can be done as step #2 also)
   -->
   <div id="clouds">
     <div class="cloud 1"></div>
     <!-- Time for multiple clouds to dance around -->
     <div class="cloud 2"></div>
     <div class="cloud 3"></div>
     <div class="cloud 4"></div>
     <div class="cloud 5"></div>
   </div>
   <!-- Thats the markup! -->
   <div class="cloud x"></div>
   <!-- That looks cool. We are done! -->
   <!-- Disregard -->
</html>
```
Infrastructure

code editor

code previewer (local)
Infrastructure

- web server
- code editor
- code previewer (local)

changes and event updates
Infrastructure

- web server
- sketch
- database server

Changes and event updates

Persistence

Code editor

Code previewer (local)
Infrastructure

- web server
- database server
- code editor
- code previewer (local)
- code previewer (remote)

Changes and event updates flow from the code editor to the web server, which is linked to the database server. The persistence of changes goes from the web server to the database server. Events and real-time events include code changes, playback calls, and code helpers.

The diagram illustrates the flow of information and the components involved in a software development environment.
Implementation: Code Changes
Implementation: Code Changes

- HTML
Implementation: Code Changes

- **HTML**

  - diffDOM: A JavaScript diffing algorithm for DOM elements

  ![Diagram](image-url)
Implementation: Code Changes

- **HTML**
  
  diffDOM: A JavaScript diffing algorithm for DOM elements

- **CSS**
Implementation: Code Changes

• HTML
diffDOM: A JavaScript diffing algorithm for DOM elements

Document A

Diff

Delta A/B

Patch

Document B

Document B'

• CSS

Document A

Replace

Document B'

Document B

• JavaScript —> still to be done! (very hard)
Implementation: Helpers
Implementation: Helpers

- Contextual helpers can facilitate the authoring process:
  a) color picker
  b) slider and
  c) angle picker
Implementation: Helpers

• Contextual helpers can facilitate the authoring process:
  a) color picker
  b) slider and
  c) angle picker

• Inlet: JavaScript plugin for CodeMirror
User Evaluation
User Evaluation

- 22 post-secondary students over 2 weeks (IFES)
User Evaluation

• 22 post-secondary students over 2 weeks (IFES)

• Exercises on how to use SVG graphics on the Web
User Evaluation

- 22 post-secondary students over 2 weeks (IFES)
- Exercises on how to use SVG graphics on the Web
- Results based on SUS framework and open-ended questions
User Evaluation

- 22 post-secondary students over 2 weeks (IFES)
- Exercises on how to use SVG graphics on the Web
- Results based on SUS framework and open-ended questions

Mean SUS score = 90.0 / Learnability score = 84.7
Final Remarks
Final Remarks

• Valuable tool to teach concepts of time on Web documents using the *Problem-Based Learning* (PBL) methodology
Final Remarks

- Valuable tool to teach concepts of time on Web documents using the *Problem-Based Learning* (PBL) methodology

- Next steps: improve current implementation based on user evaluation
Final Remarks

• Valuable tool to teach concepts of time on Web documents using the Problem-Based Learning (PBL) methodology

• Next steps: improve current implementation based on user evaluation

• Add support code snippets in SMIL Timesheets and Time Style Sheets (TSS)
Final Remarks

• Valuable tool to teach concepts of time on Web documents using the Problem-Based Learning (PBL) methodology

• Next steps: improve current implementation based on user evaluation

• Add support code snippets in SMIL Timesheets and Time Style Sheets (TSS)

• We need a fresh new look into the standardization of time-based APIs on the Web
Final Remarks

• Valuable tool to teach concepts of time on Web documents using the Problem-Based Learning (PBL) methodology

• Next steps: improve current implementation based on user evaluation

• Add support code snippets in SMIL Timesheets and Time Style Sheets (TSS)

• We need a fresh new look into the standardization of time-based APIs on the Web

• Offer mechanisms to control and simulate the behavior of elements over time (e.g. temporal visualization)
Thanks!

Rodrigo Laiola Guimarães
rodrigo@laiola.com.br