

DeployGround:

A Framework for Streamlined Programming
from API Playgrounds to Application Deployment

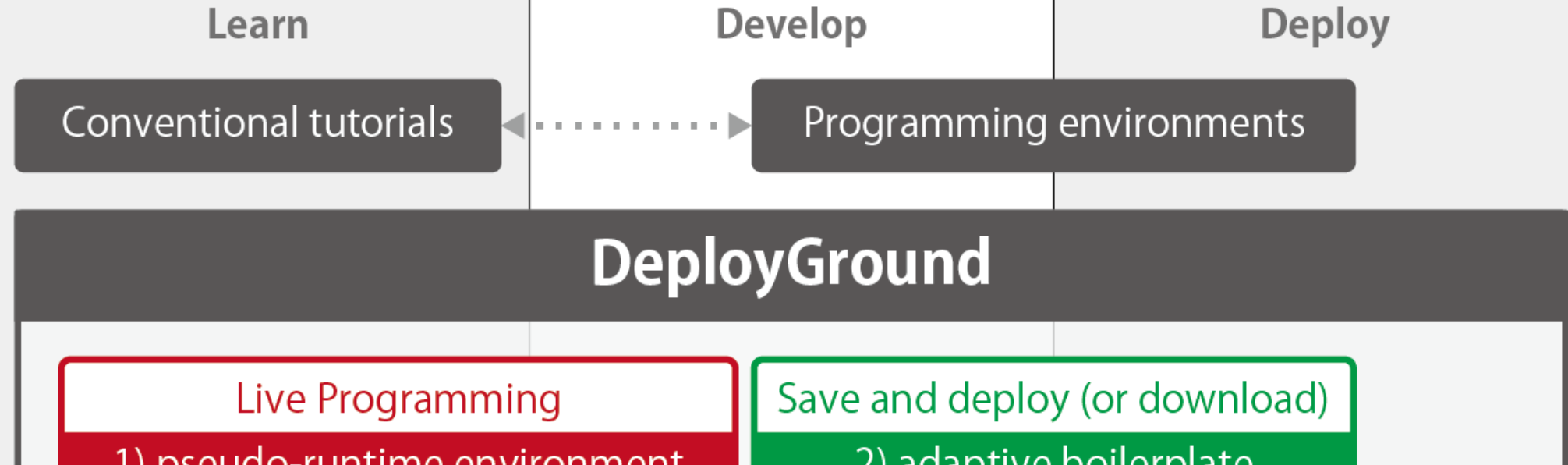
Jun Kato, Masataka Goto

National Institute of Advanced Industrial Science and Technology (AIST), Japan

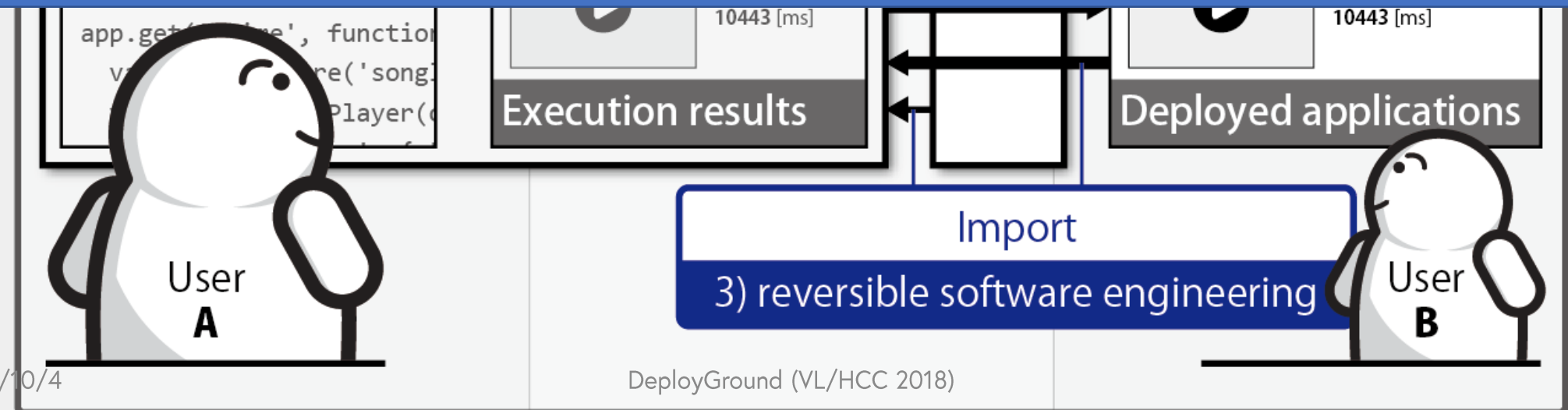
VL/HCC 2018 Short paper (10 min talk)



3) reversible software engineering



Let me do the demo first!



Coding tutorials and references

- Much work on creating tutorials in the context of HCI
[Chi et al., UIST '12] [Lafreniere et al., CHI '13] [Chi et al., UIST '13] [Kim et al., CHI '14]
- Only a handful of work on creating coding tutorials
[Harms et al., IDC '13] [Head et al., VL/HCC '15] [Gordon et al., VL/HCC '15]

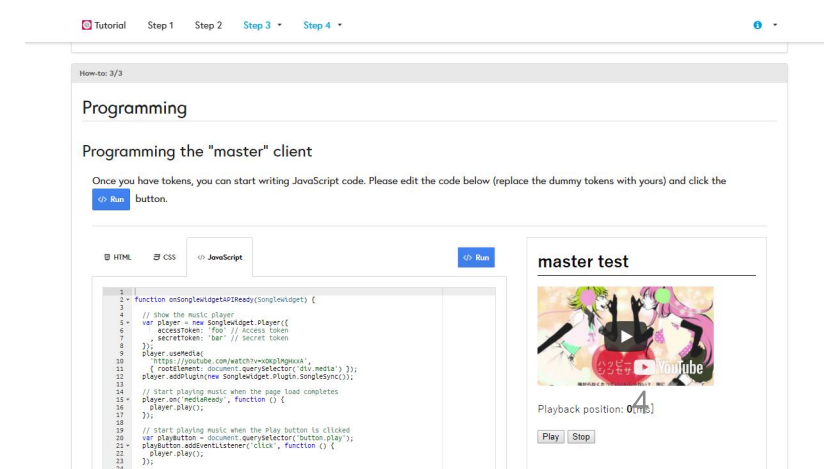
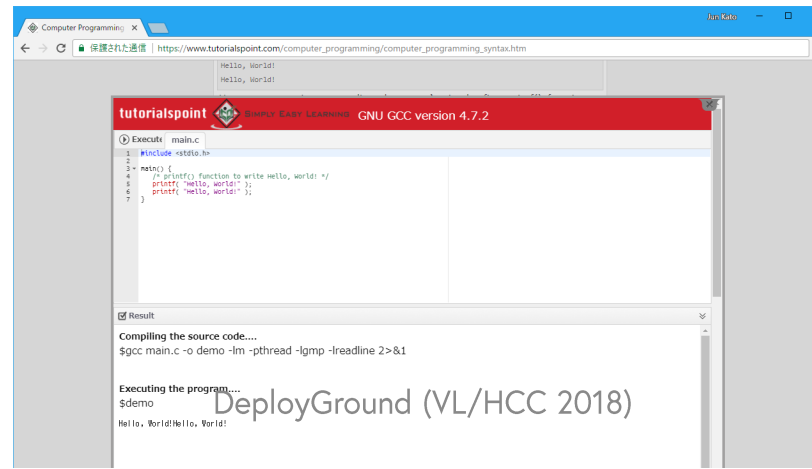
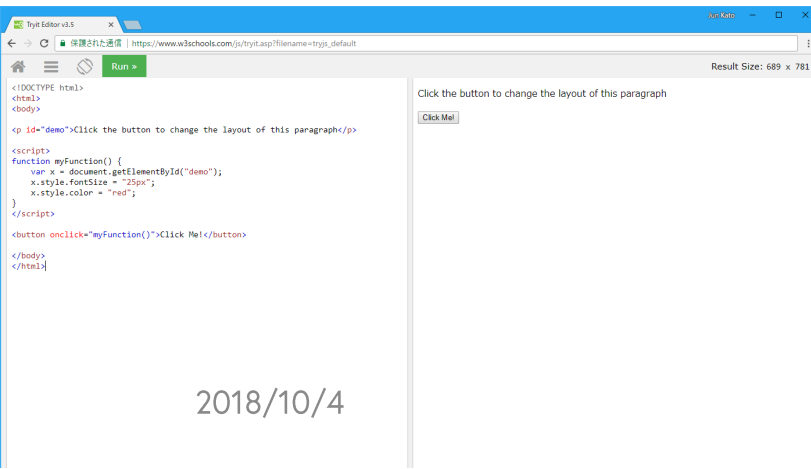
See <https://junkato.jp/deployground> for the complete list of references.

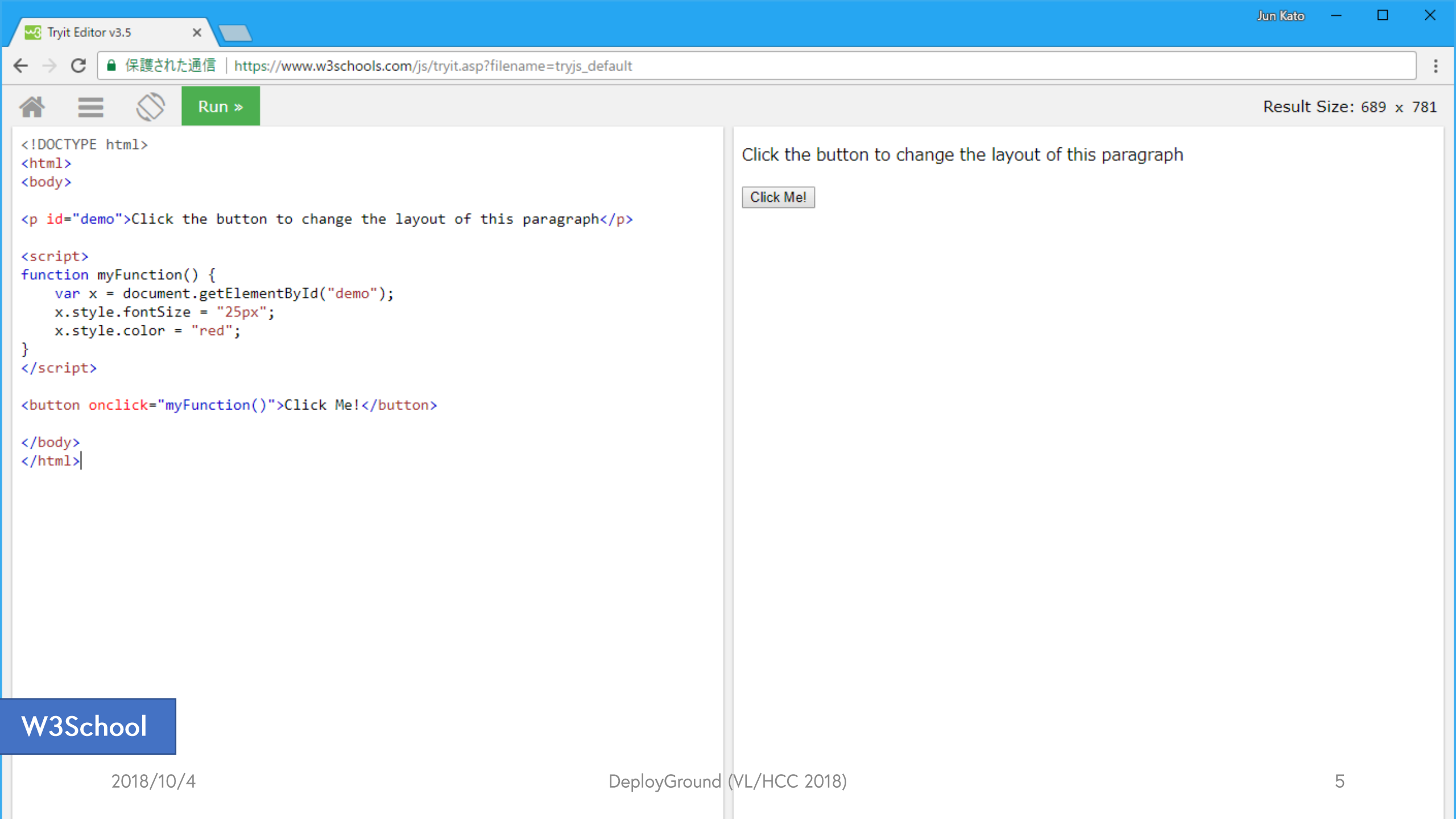
Contributions:

- A framework to create interactive web-based coding tutorials
- Its concrete implementation techniques

API documentations and tutorials

- Concrete usage information is beneficial
[Robillard, 2009] [Hou et al., ICPC '11] [Robillard et al., 2011] [Wang et al., MSR '13]
- Executable example codes are especially beneficial
[Subramanian et al., ICSE '14]
- Some documentations and tutorials provide "playgrounds"
[Khan Academy] [TypeScript Playground] [Vimeo API Playground] [W3School] [tutorialspoint] ...





Hello, World!

Hello, World!

tutorialspoint



SIMPLY EASY LEARNING

GNU GCC version 4.7.2

▶ Execute main.c

```
1 #include <stdio.h>
2
3 main() {
4     /* printf() function to write Hello, World! */
5     printf( "Hello, World!" );
6     printf( "Hello, World!" );
7 }
```

☒ Result

Compiling the source code....

\$gcc main.c -o demo -lm -pthread -lgmp -lreadline 2>&1

Executing the program....


demo

, World!Hello, World!

How-to: 3/3

Programming

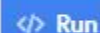
Programming the "master" client

Once you have tokens, you can start writing JavaScript code. Please edit the code below (replace the dummy tokens with yours) and click the  Run button.

HTML

CSS

JavaScript

 Run

```
1
2 function onSongWidgetAPIReady(SongWidget) {
3
4     // Show the music player
5     var player = new SongWidget.Player({
6         accessToken: 'foo' // Access token
7         , secretToken: 'bar' // Secret token
8     });
9     player.useMedia(
10         'https://youtube.com/watch?v=xOKpLMgHxxA',
11         { rootElement: document.querySelector('div.media') });
12     player.addPlugin(new SongWidget.Plugin.SongSync());
13
14     // Start playing music when the page load completes
15     player.on('mediaReady', function () {
```

Songle Sync tutorial (DeployGround-based tutorial)

2018/09/4

```
20 var playButton = document.querySelector('button.play');
21 playButton.addEventListener('click', function () {
22     player.play();
23 });
24
```

DeployGround (VL/HCC 2018)

master test



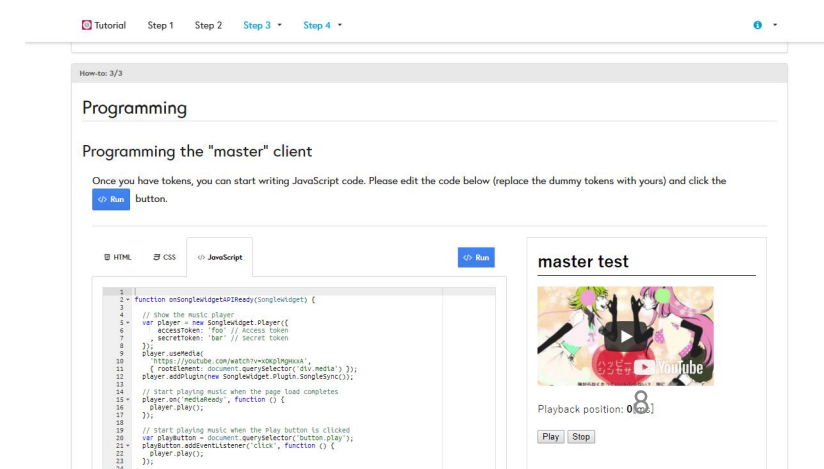
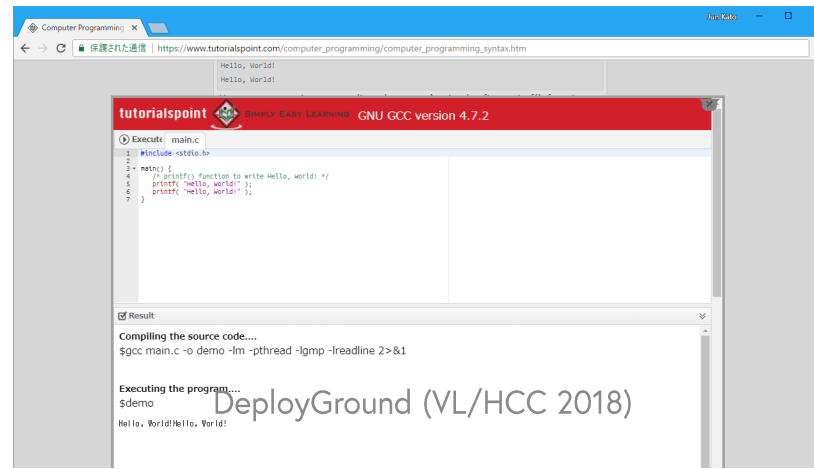
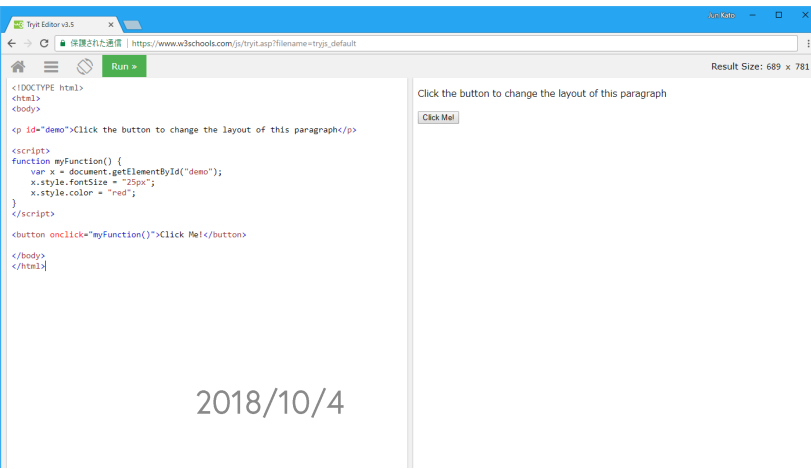
Playback position: 0[ms]

Play

Stop

API Playgrounds

- A part of coding tutorials (API documentations or tutorials)
- A code editor and its output sit next to each other
- The output can be interactively updated upon the user's request



Limitations of existing tutorials

- Learning APIs is supported well with the interactive playgrounds
meanwhile...
- The programmer needs to leave the tutorial at some point
- S/he needs to re-start the development in their own environment.

Support for seamless post-learning experience is in need

Detailed limitations of existing tutorials

- Ephemeral code
- Toy sandbox OR expensive sandbox
- No support for deployment
- Little social interaction

What are these and how can we address them?

1

```
<!DOCTYPE html>
<html>
<body>

<p id="demo">Click the button to change the layout of this paragraph</p>

<script>
function myFunction() {
  var x = document.getElementById("demo");
  x.style.fontSize = "25px";
  x.style.color = "red";
}
</script>

<button onclick="myFunction()">Click Me!</button>

</body>
```

Click the button to change the layout of this paragraph

Click Me!

Iframe element

2

tutorialspoint



SIMPLY EASY LEARNING

GNU GCC version 4.7.2

Execute

main.c

```
1 #include <stdio.h>
2
3 main() {
4     /* printf() function to write Hello, World! */
5     printf( "Hello, World!" );
6     printf( "Hello, World!" );
7 }
```

☒ Result

Compiling the source code....

```
$gcc main.c -o demo -lm -pthread -lgmp -lreadline 2>&1
```

Executing the program....

demo

, World!Hello, World!

**Virtual
Machine
on the
server**

tutorialspoint

Programming

Programming the "master" client

Once you have tokens, you can start writing JavaScript code. Please edit the code below (replace the dummy tokens with yours) and click the

</> Run button.

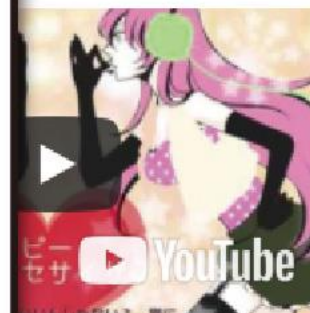
HTML

CSS

</> JavaScript

```
1
2 function onSongleWidgetAPIReady(SongleWidget) {
3
4     // Show the music player
5     var player = new SongleWidget.Player({
6         accessToken: 'foo' // Access token
7         , secretToken: 'bar' // Secret token
8     });
9     player.useMedia(
10         'https://youtube.com/watch?v=xOKpIMgHxxA',
11         { rootElement: document.querySelector('div.media') });
12     player.addPlugin(new SongleWidget.Plugin.SongleSync());
13
14     // Start playing music when the page load completes
15     player.on('mediaReady' function () {
```

Interpreter
on the
web
browser

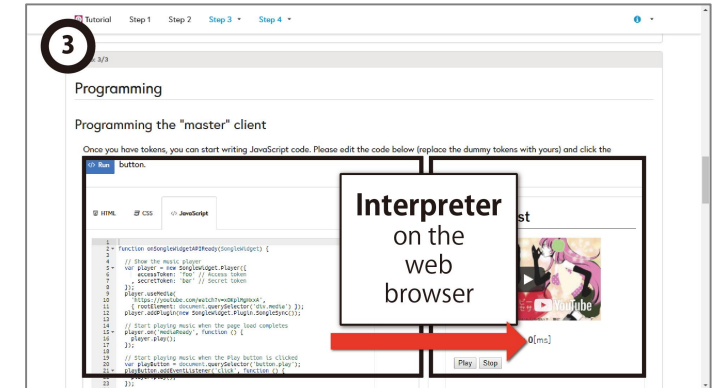
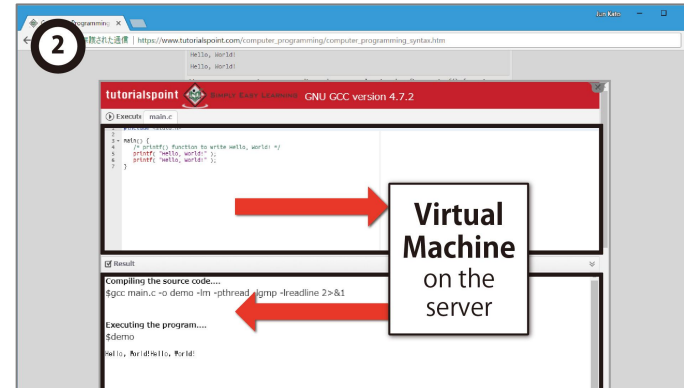
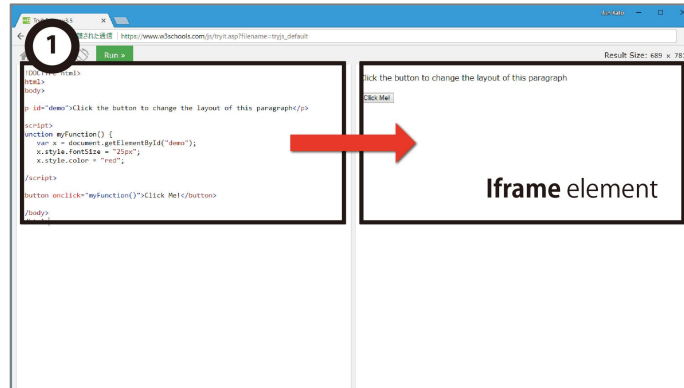


0[ms]

Play Stop

Songle Sync tutorial (DeployGround-based tutorial)

"Playground" implementations



Performance

Browser-native

Huge latency

A little overhead

Debuggability

Very low

Low

High

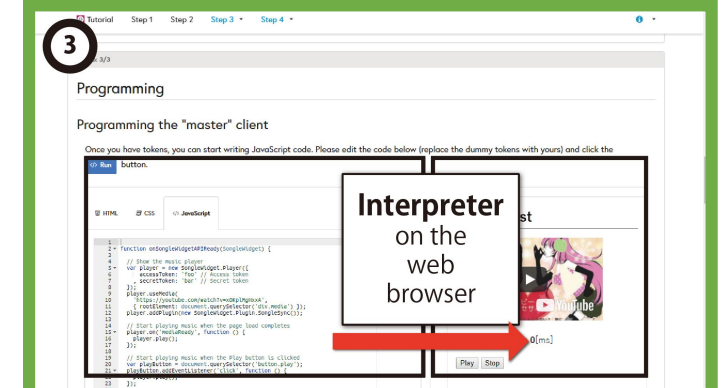
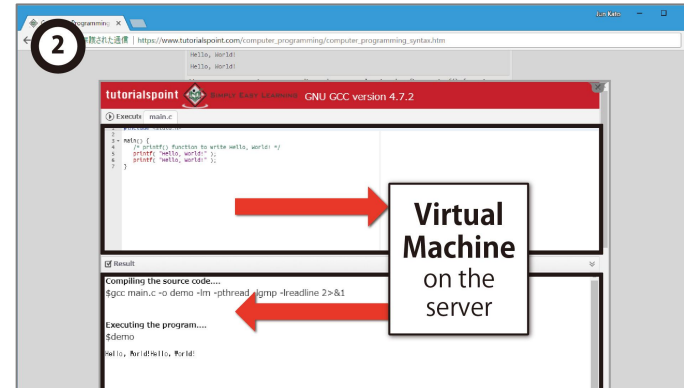
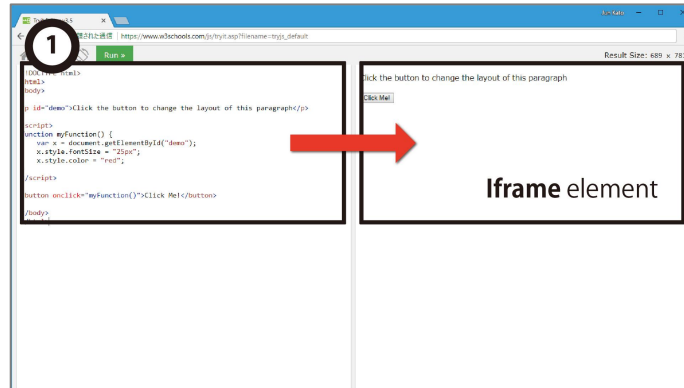
Supported APIs:

Complete (browser-based APIs)

Complete (CUI-based APIs)

Partial (any emulatable APIs)

"Playground" implementations



Performance

Browser-native

Huge latency

Debuggability

Very low

Low

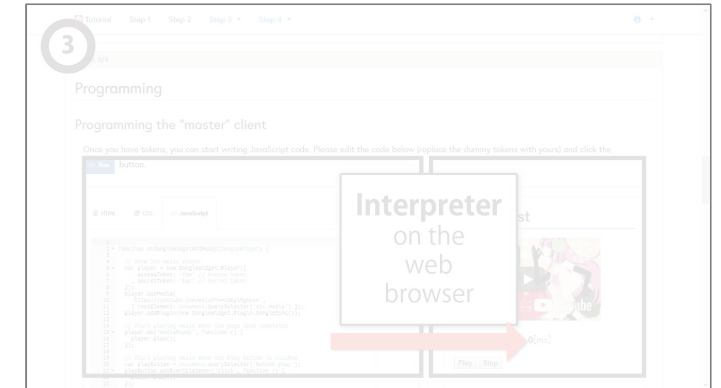
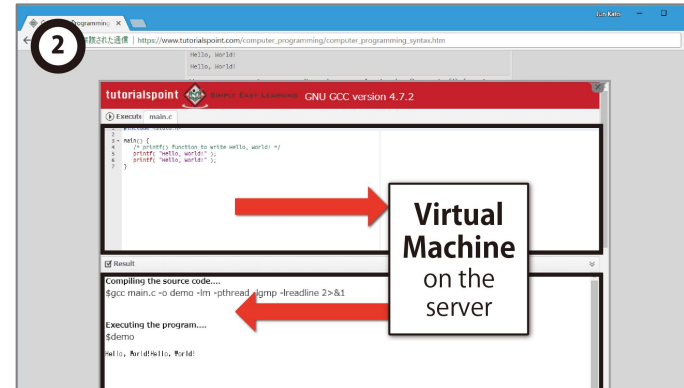
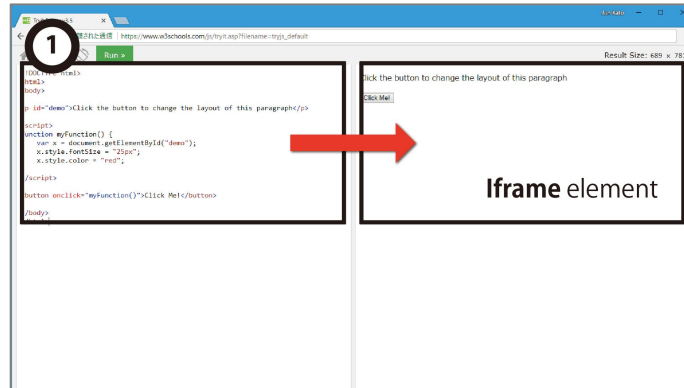
Supported APIs:

Complete (browser-based APIs)

Complete (CUI-based APIs)

DeployGround

Ephemeral code?



Download each
code editor
content as a file

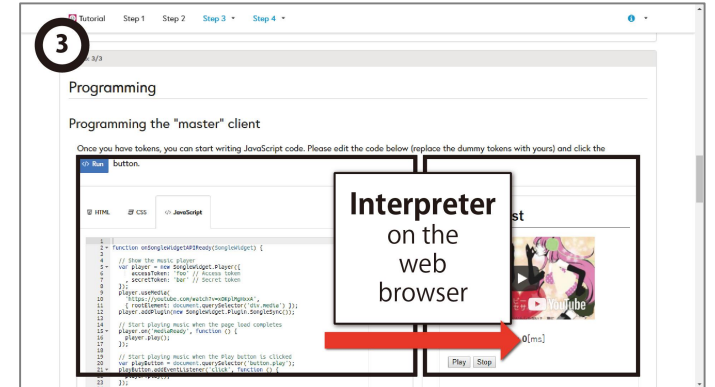
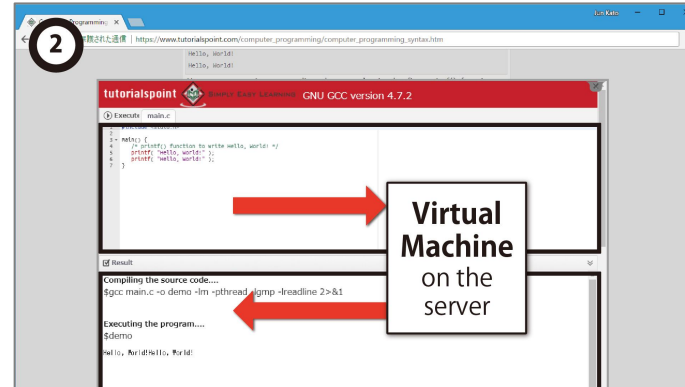
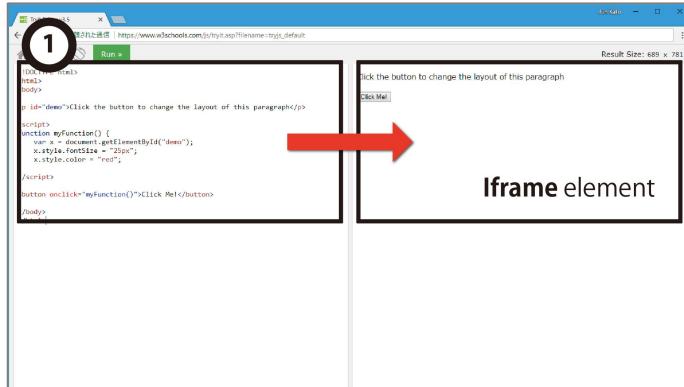
VM sessions
cannot be (easily)
exported

A little overhead

High

Partial (any emulatable APIs)

Ephemeral code?

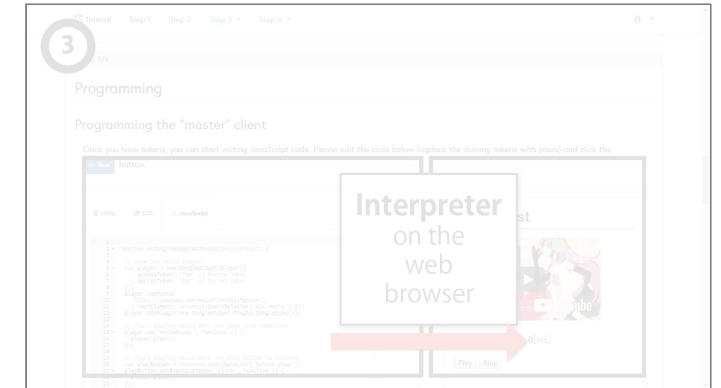
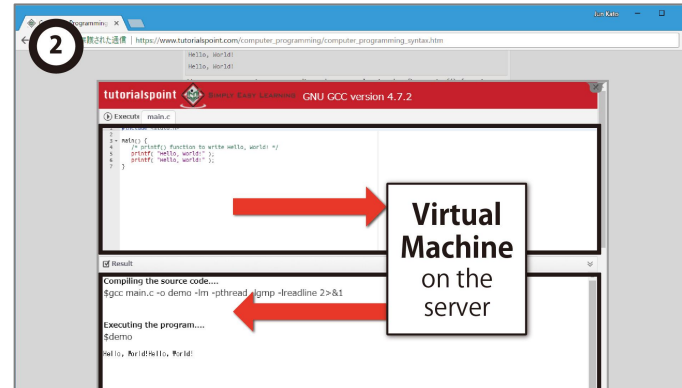
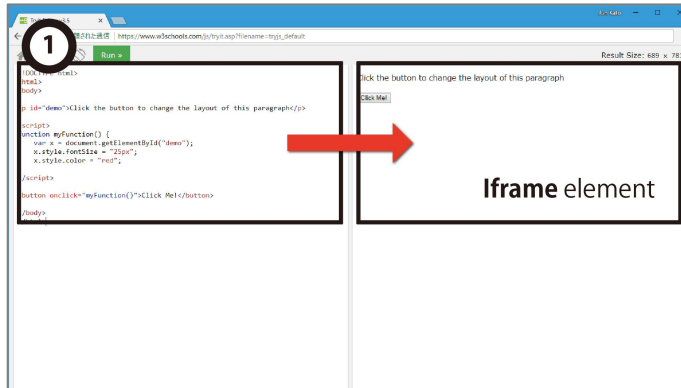


Download each
code editor
content as a file

VM sessions
cannot be (easily)
exported

Save the entire
workspace
as a GitHub repo

Toy sandbox OR expensive sandbox?



Toy:
cannot used for non-
browser languages

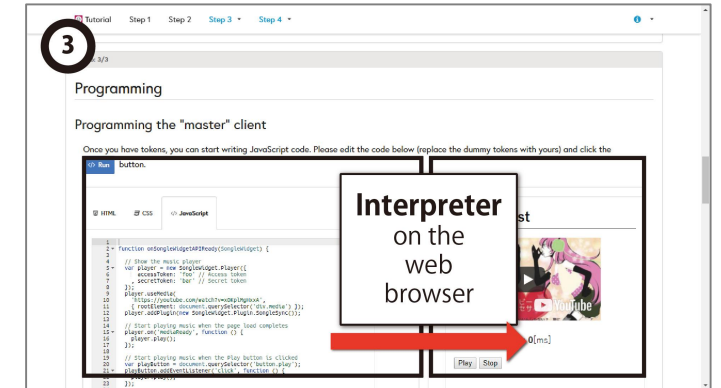
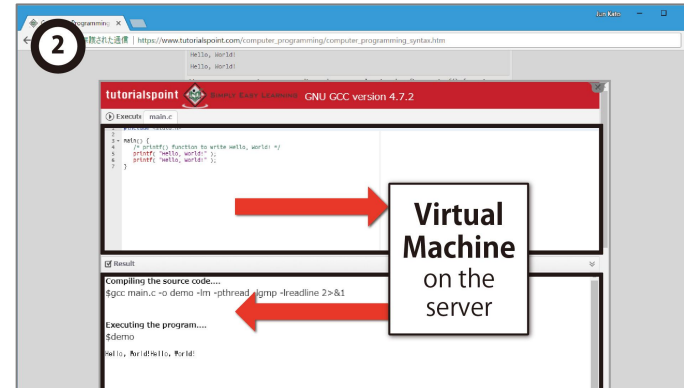
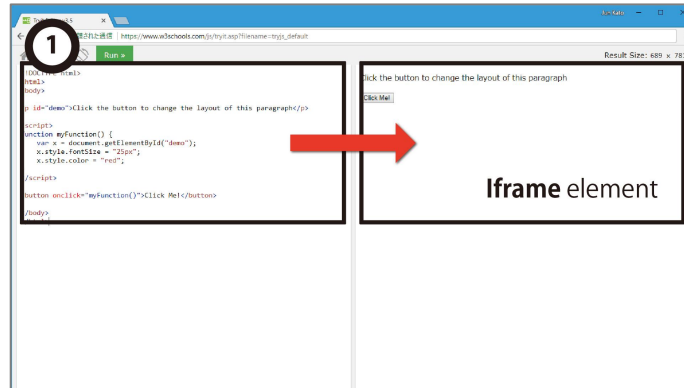
Expensive:
requires lots of
server-side resources

A little overhead

High

Partial (any emulatable APIs)

Toy sandbox OR expensive sandbox?

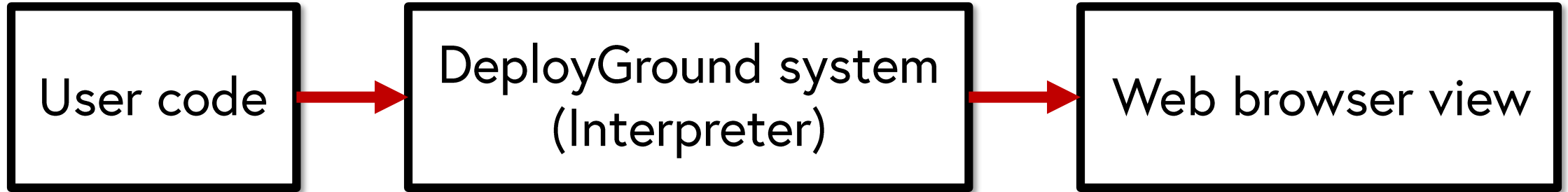


Toy:
cannot used for non-
browser languages

Expensive:
requires lots of
server-side resources

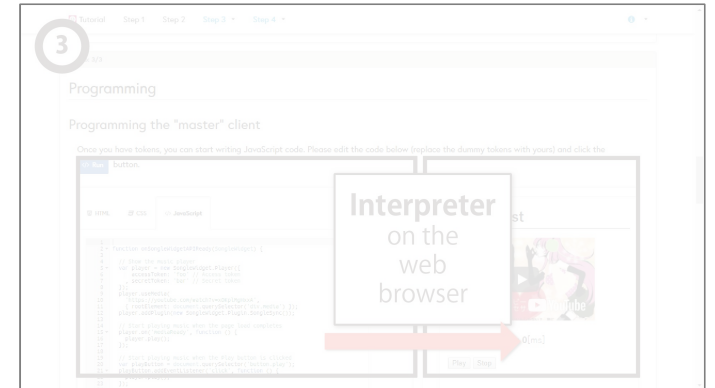
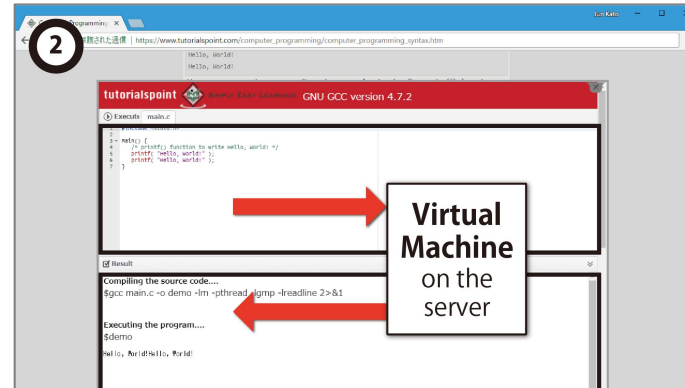
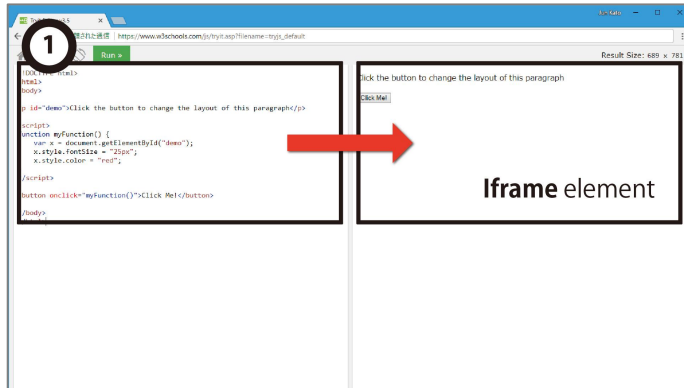
Flexible:
can potentially
support any APIs

Pseudo runtime environment for making a flexible sandbox



- requires manual implementation of the emulation layer
- yet has potential to emulate anything that can be represented (visualized) on a web browser
- e.g.,
 - Node.js-based web servers: usually requires dedicated domain names
 - Physical computing devices: usually requires purchasing modules

No support for deployment?



HTML with JS files cannot be directly opened with modern web browsers

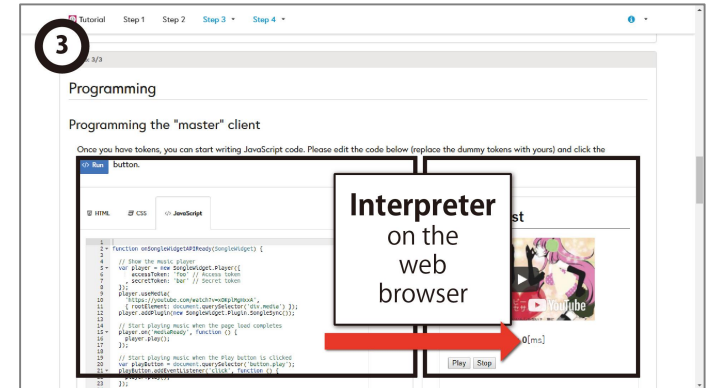
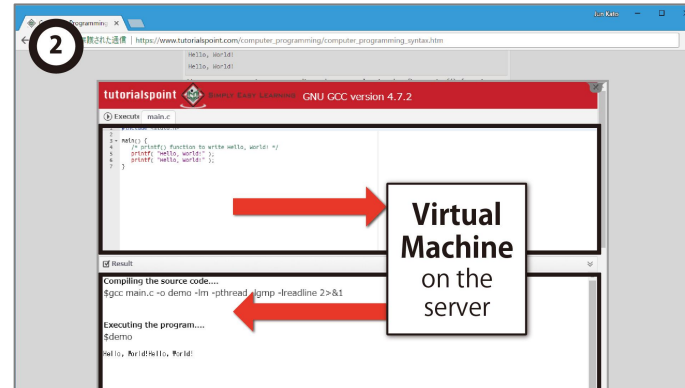
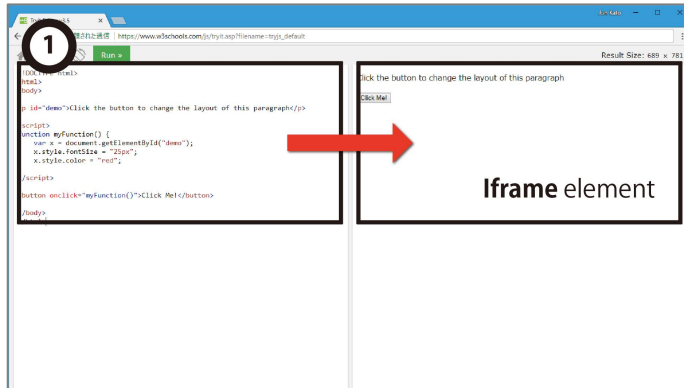
In more complex cases (e.g. Node.js-based projects), only instructions are presented

A little overhead

High

Partial (any emulatable APIs)

No support for deployment?



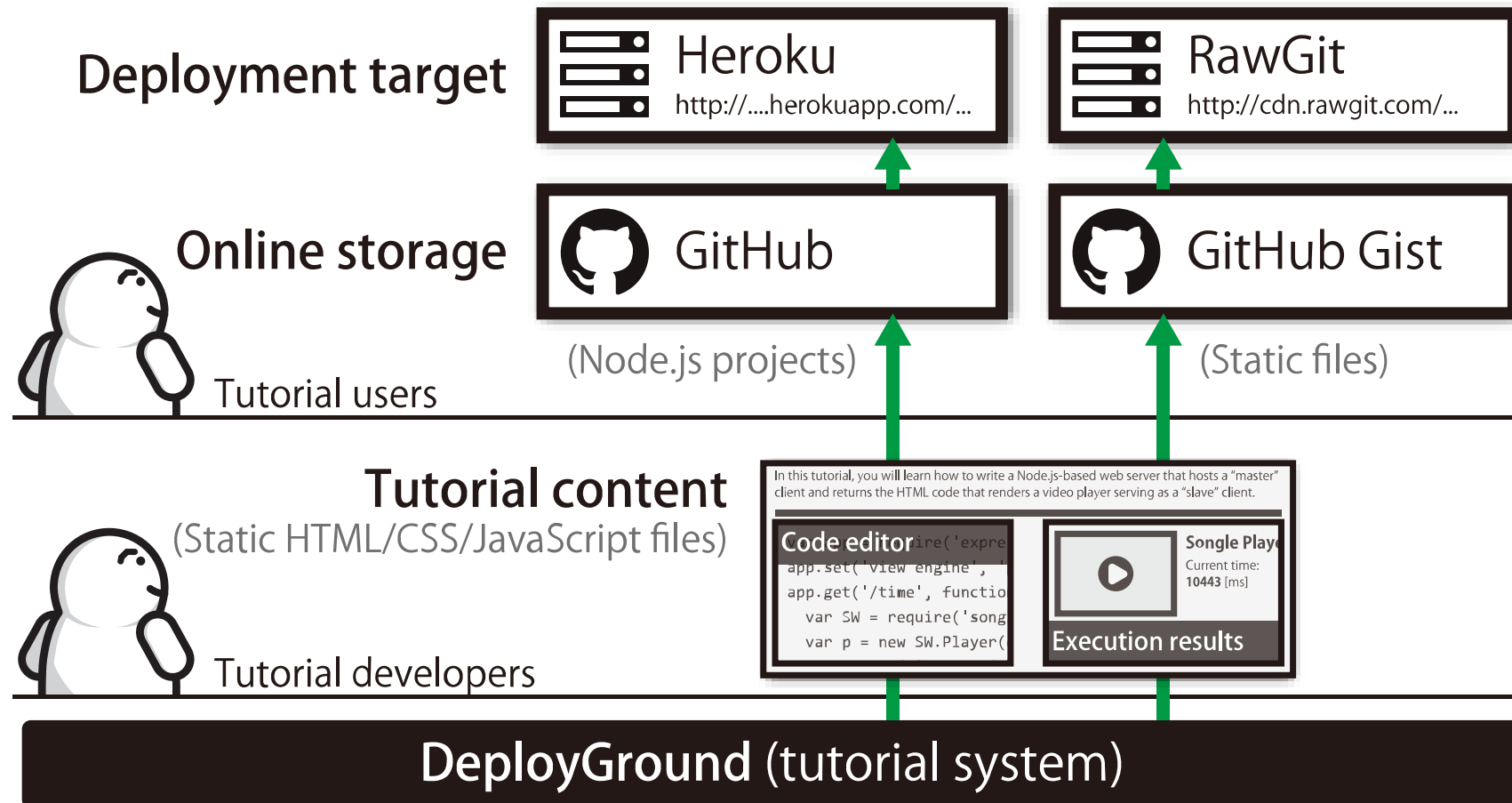
HTML with JS files cannot be directly opened with modern web browsers

In more complex cases (e.g. Node.js-based projects), only instructions are presented

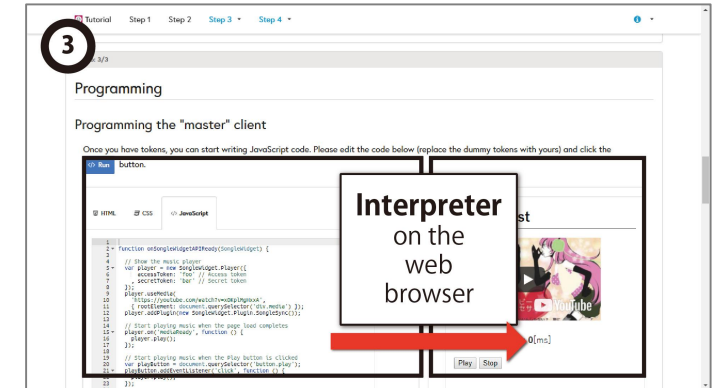
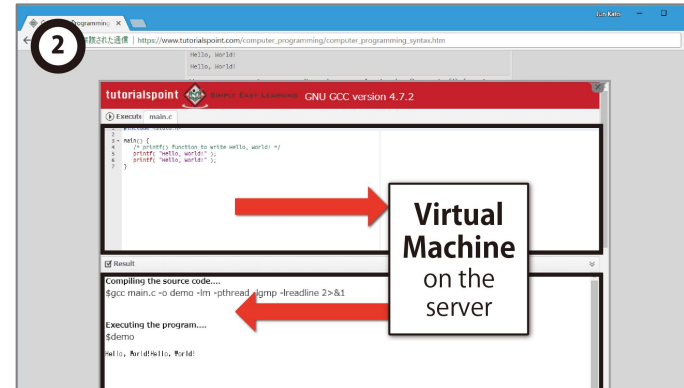
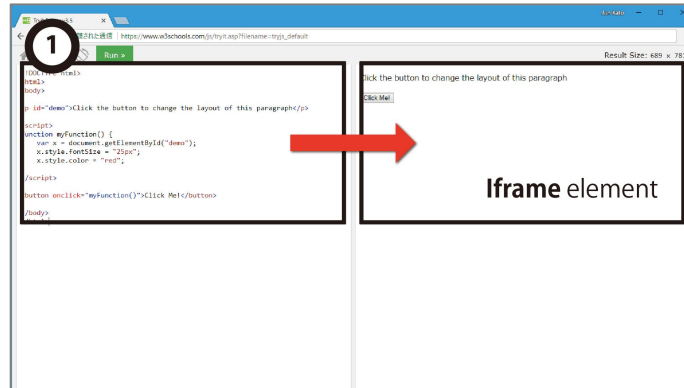
Deploy to GitHub
Gist or Heroku

Adaptive boilerplate

for exporting files as an executable project



Little social interaction?

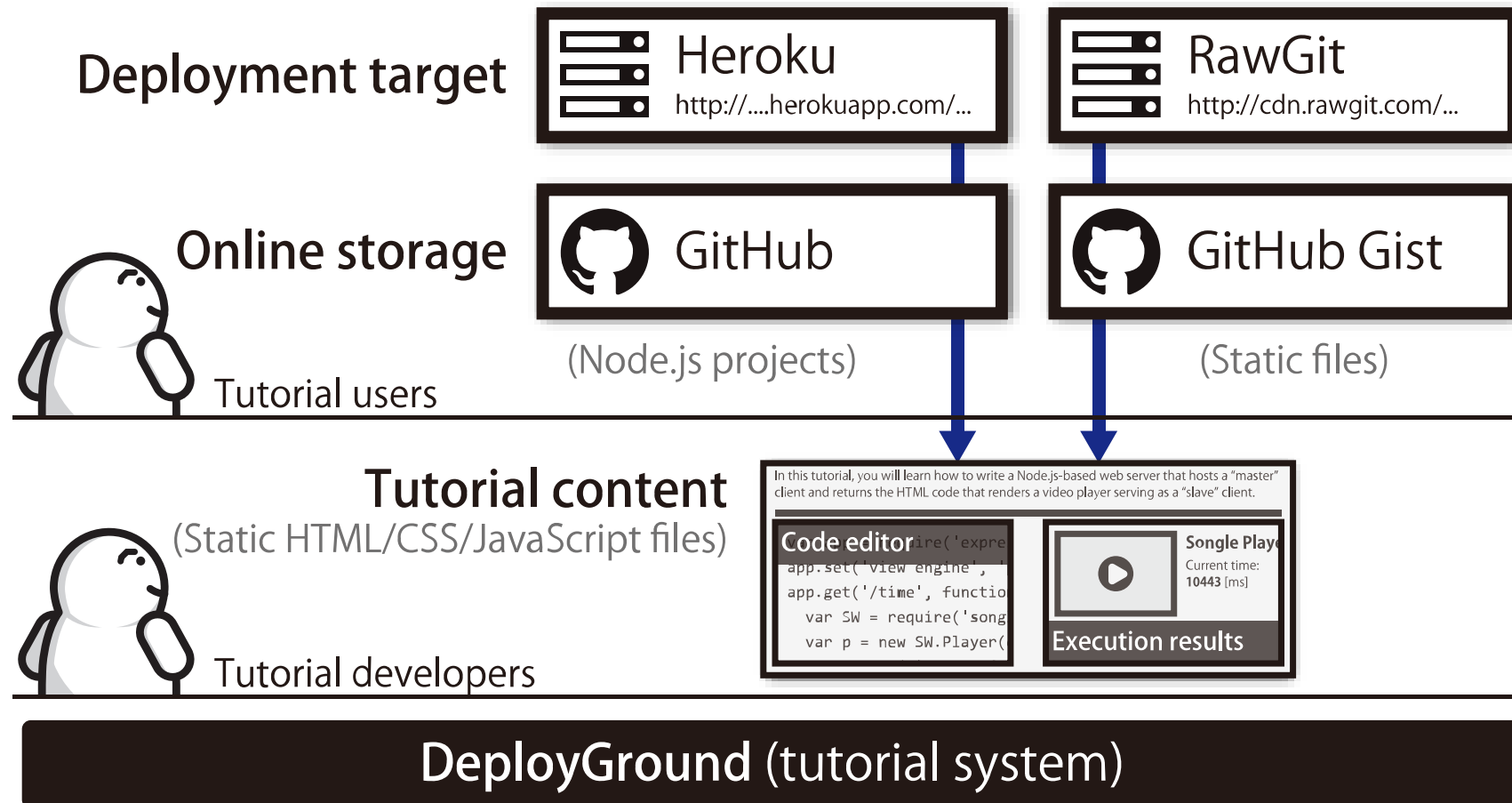


"Achievement unlocked!" kind of posts can be potentially made on social networking services

Codebases and apps can be shared instantly

Reversible software engineering

to use people's outcome as educational resource



Conventional tutorials

Programming environments

DeployGround

Live Programming

1) pseudo-runtime environment

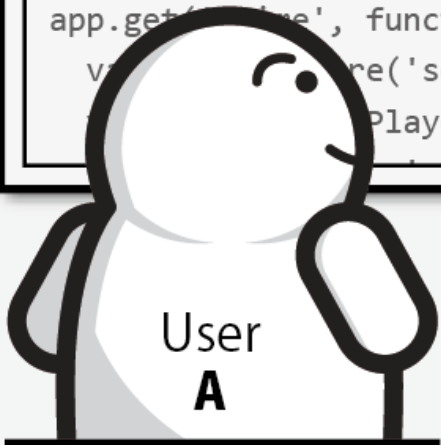
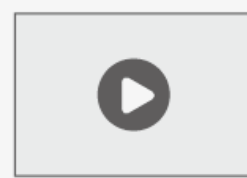
Save and deploy (or download)

2) adaptive boilerplate

In this tutorial, you will learn how to write a Node.js-based web server that hosts a "master" client and returns the HTML code that renders a video player serving as a "slave" client.

Code editor

```
var express = require('express');
app.set('view engine', 'pug');
app.get('/', function(req, res) {
  res.render('song');
});
```

User
A

Songle Player

Current time:
10443 [ms]

Execution results



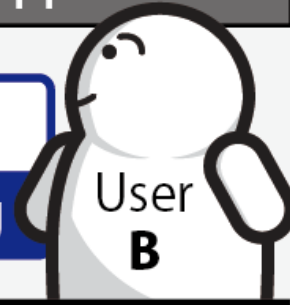
Songle Player

Current time:
10443 [ms]

Deployed applications

Import

3) reversible software engineering

User
B

Learn

Develop

Deploy

Conventional tutorials

Programming environments

DeployGround

Live Programming

1) pseudo-runtime environment

Save and deploy (or download)

2) adaptive boilerplate

←from learning Streamlined support to deployment→

```
app.get('/tree', function  
  view('tree', {  
    song: 'Song of the  
    Player(
```

User
A

10443 [ms]

Execution results

10443 [ms]

Deployed applications

Import

3) reversible software engineering

User
B

Preliminary user feedback

Refer to the paper

Asked 3 software engineers and 2 researchers to answer prequestionnaire and try out the tutorial

Asked 24 university students to form 6 groups and prototype applications in 2 days

- All of them successfully benefited from the framework
- Potential applications: APIs with expensive initial cost to try out
- Requests for more detailed views on save/deploy features
- Emulation is imperfect; more diverse examples are demanded

DeployGround framework

- **Pseudo runtime environment** enables live programming experience.
Learners can enjoy testing the APIs. They accumulate their codebase during the tutorial.
- **Adaptive boilerplate** brings the experience out of the sandbox.
In the end, they get the archived project files. The tutorial even helps them deploy the project.
- **Reversible software engineering** allows to gain benefits from social coding.
All of the above experiences can be easily shared on the web, helping the other learners.