Dart Code Lab

Kyoto GDG

2012/09/29

目 次

1 Introduction

This codelab will help you build and run a simple chat application using Dart. Along the way, you will learn:

- The fundamentals of the Dart programming language
- How to use the Dart HTML libraries
- Bi-directional communication with WebSockets
- The basics of Dart Editor
- How to find answers from the online resources

1.1 Prerequisites

This codelab assumes you are familiar with fundamental web programming. You should know the basics of HTML, CSS, and JavaScript. You should also have experience with a programming language such as Java, C#, or JavaScript.

This codelab assumes you have watched the following video tutorials:

- Google I/O 101: Introduction to Dart with Seth Ladd
- Google I/O 101: Dart Editor with Devon Carew

1.2 Installs

This codelab requires the Dart Editor. You can find builds of the editor for Mac, Windows, and Linux at `http://dartlang.org/editor/`.

1.3 Additional materials

This codelab provides easy to follow, step-by-step instructions. However, you 'll find it quite useful to have the following online resources loaded up and ready to access.

- `http://api.dartlang.org` The Dart API docs list the functions, classes, and methods for all the libraries.
- `http://www.dartlang.org/docs/language-tour/` The Dart Language Tour is your high-level guide through the Dart language.
- `http://www.dartlang.org/docs/library-tour/` The Dart Library Tour covers most of the bundled libraries of the Dart platform.
- `http://synonym.dartlang.org/` Know JavaScript? This is your resource! Translate common idioms, patterns, and snippets from JavaScript to Dart.

2 Step 1: Set up your environment

Dart offers better productivity through powerful and helpful tools. At the center of the toolchain is the Dart Editor, a lightweight text editor that understands how to analyze, run, and debug Dart apps. The editor works with the Dart SDK and Dartium (a build of Chromium with the Dart VM) to give you an integrated experience.

2.1 Objectives

1. Install Dart Editor 2. Send feedback to the editor team 3. Run the sample clock Dart app 4. Learn about Dartium

2.2 Walkthrough

2.2.1 Install the Dart Editor

To get your environment set up, plug in the provided USB. Open the USB drive and find the `editor/` directory inside. Copy over the correct Dart Editor version for your OS/bit combination directory to your machine, and unzip it.

[Image]

Open up the newly unzipped `dart/` directory, and double click the executable:

- DartEditor.app (Mac)
- DartEditor (Linux)
- DartEditor.exe (Windows)

[Image]
When Dart Editor first opens, the Welcome window appears.
[Image]

2.2.2 Use the Send Feedback button

The Dart Editor team really appreciates feedback. The easiest way to let them know your thoughts is to use the Send Feedback button in the upper right of the editors toolbar.

Locate the Send Feedback button and click on it.

[Image]

The Dart Editor Feedback dialog allows you to share bugs and requests directly with the editor team, as well as the larger Dart team. Feel free to send us any and all comments, especially during this codelab. Well turn your suggestions into bug reports and feature requests3 as appropriate.

[Image]

2.2.3 Run the Clock sample

(If the feedback window is still open, close it now.)

Time to run a Dart app!

Click on the Clock sample from the Welcome window, which copies the Clock sample into Dart Editor and sets up a new project for you.

[Image]

Tip: Cant find the Welcome view? You can go to Tools, Welcome Page to display it again.

[Image]

The Files view shows all of the files in the Clock sample, including all Dart files, the HTML file that hosts the app, as well as all of the images and CSS files. clock.dart is a Dart file that defines the "clock" library,

and includes the main() method for the sample. The clock.dart file is automatically opened in the editor.

[Image]

Ensure the clock.dart file is selected and highlighted. Click the Run button in Dart Editor, which launches the application by loading Dartium and pointing it to the clock.html file.

[Image]

Here is the Clock sample app running inside Dartium. Congratulations, you are running your first Dart app!

[Image]

2.2.4 Dartium

Dartium is Chromium with an embedded Dart virtual machine (VM). Even though Dart compiles to JavaScript, you can speed up the "edit, reload" development cycle by running Dart apps directly in the browser.

To verify that Clock is running in Dartium, right-click in the browser and select Inspect Element.

[Image]

The Elements tab should be selected by default, and clock.dart should be listed as the script that is being run.

[Image]

2.2.5 Debug with the Dart Editor

With Dart running directly in Dartium, the editor has debugging support for Dart applications. Set a breakpoint by double clicking in the left hand gutter in Dart Editor on first call to setDigits() inside the updateTime() method in clock.dart, line 66. With the breakpoint set (you will see a little blue dot in the gutter), click the Run button again.

[Image]

Notice how the program stop, and the Debugger view opens in the Dart Editor. To continue the program without leaving the debugger, click the Resume button (the green arrow in the Debugger view). The updateTime() method is called every 1000ms, therefore the breakpoint will be hit again within a second.

[Image]

The Debugger view on the right hand side allows you to see which processes are running and what values are in scope at the breakpoint. Hover over the now field, the value will be displayed in a tooltip.

[Image]

To terminate the debugger, click on the red square in the Debugger view. This will also stop the application.

[Image]

2.3 Advanced

Load and launch the other samples. In the Clock sample, try changing the ball velocity or the gravity. Start with lines 12 and 117 in balls.dart. Set more breakpoints and inspect the values by opening up the variables.

[Image]

謝辞

この文書の翻訳にあたり,