calgary scientific

PureWeb® STK 4.0

Quick Start Guide: HTML5 Client

About Calgary Scientific

Calgary Scientific Inc. is dedicated to providing advanced visualization, web enablement, and mobility enhancement solutions to industries looking for secure access and use of their data or graphics intensive applications, while using their existing systems. Visit www.calgaryscientific.com for more information.

Notice

Your Responsibility for Your System’s Security

You are responsible for the security of your system. Product administration to prevent unauthorized use is your responsibility. Your system administrator should read all documents provided with this product to fully understand the features available that reduce your risk of incurring charges for unlicensed use of Calgary Scientific products.

Trademarks

Released by

Calgary Scientific Inc. www.calgaryscientific.com.

Document Version: PW4.0_HTMLClient_QSG_07-2013_v1.000.00

calgary scientific

Table of Contents

Preface 4

Chapter 1 **Introduction** 5

Chapter 2 **Deployment Procedures** 7

Prerequisites 7
Setting the Environment Variables 8
Installing PureWeb 9
Deploying the Sample Applications 9

Chapter 3 **Applications Overview** 10

Launching the Applications 10
Exercising the Functionality 11
- Scribble 11
- Asteroids 12

Diagnostics Panel 13
Sharing an Application 14

Chapter 4 **Exploring the Code** 15

Scribble Client Code Pointers 15
- Page structure 15
- Page Elements Specific to PureWeb 16
- Interacting with the PureWeb Service Application 17
- Custom Touch Events 18

Asteroids Client Code Pointers 19
- Page Elements Specific to PureWeb 19

Chapter 5 **Debugging** 21

Preface

Welcome to the *PureWeb STK 4.0 Quick Start Guide: HTML5 Client*, a document that provides instructions to install the PureWeb STK, and to build, run and troubleshoot sample PureWeb HTML5 client applications.

Intended Audience

This document is primarily intended for software developers who plan to install and use the PureWeb STK to develop PureWeb HTML5 clients. It has been written with the assumption that you have a working knowledge of the following:

- HTML
- CSS
- JavaScript

Making Comments on This Document

If you especially like or dislike anything about this document, feel free to e-mail your comments to techpubs@calgaryscientific.com.

Contacting Calgary Scientific Support

Use one of the methods in the table below to contact Calgary Scientific support.

Web Site	E-Mail
support.getpureweb.com	support@getpureweb.com

Chapter

1 Introduction

PureWeb is a platform that enables the rapid transformation of enterprise software into cloud-ready, web and mobile applications.

A PureWeb application consists of three main components: a service application, a client application, and the PureWeb server which integrates the two.

The PureWeb STK (software transformation kit) is a set of tools that provides developers everything they need to implement the PureWeb solution in their own applications, in particular service and client APIs, documentation and API reference material, and sample applications in each supported programming language.

The sample applications illustrate key concepts of PureWeb enablement; they can be used by developers as a starting point to create their own PureWeb-enabled applications.

The PureWeb Quick Start Guide series is intended to get developers up and running as quickly as possible, using these sample applications as models.

As its name indicates, the current document, *PureWeb STK Quick Start Guide: HTML5 Client*, focuses on the HTML5 client. The guide provides instructions to install the PureWeb STK, and to build, run and troubleshoot a sample client.

You can choose from two HTML5 sample clients:

- Scribble: allows users to draw lines on a canvas, change line color, clear the canvas and collaborate on a drawing with another user.
- Asteroids: a PureWeb-enabled implementation of the classic video game with a twist -- it supports both single-player and two-player modes; each player independently controls his or her own ship.

Note: The PureWeb HTML5 STK Asteroids sample is intentionally agnostic of any third-party library, and uses standard HTML and JavaScript, which should work in most modern browsers.

The Scribble sample uses the JQuery JavaScript library to illustrate how to implement custom touch gestures on HTML5 mobile clients. In your own application, you can use a different library other than JQuery if you prefer.

Before working with the sample HTML5 client, you will need to build the sample service; instructions for doing so can be found in the relevant *Quick Start Guide* for the service programming language you plan to work with: C++, C# or Java for Scribble, or Java for Asteroids.

Recommended Reading

This *Quick Start Guide* is intended as an introduction, and it does not provide in-depth information. It should be used in conjunction with the other guides of the PureWeb STK documentation suite, in particular:

- *Installation Guide*
- *Developer's Guide*
- *Server Administration Guide*
- HTML5 client API reference material

To access these references in a Windows operating system, press the `Windows` key on the keyboard to display the **Start** menu (**Start** screen in Windows 8), then type "pureweb" as the search term.

Chapter

2 Deployment Procedures

In this chapter, you will find prerequisite information as well as instructions to install the PureWeb STK, and build the sample HTML5 clients.

For more detailed instructions, such as how to handle a side-by-side installation of the STK, please refer to the complete *PureWeb STK Installation Guide*.

Prerequisites

To install the PureWeb STK, you will need:

- Java JDK 1.7
- C++ Runtime, either of: 2008 SP1 ATL x86/x64 or 2010 SP1 x86/x64; here's a download link to the 2010 version: http://www.microsoft.com/en-us/download/details.aspx?id=5555
- A Windows 64-bit operating system, either of: Windows 7, Windows 8, Windows Server 2008 R2, or Windows Server 2012

Note: Although only 64-bit platforms are supported for application development, you can target the applications developed using the PureWeb STK to deploy on either a 32-bit or a 64-bit platform.

To build and run the sample HTML5 clients (either Scribble or Asteroids), you will need:

- a text editor
- an HTML5 compliant browser

Note: HTML5, by its very nature, is intended to work in many different browsers without the need of additional plug-ins. However, each browser has its own quirks, and for this reason not all browsers are officially supported by PureWeb.

For the most up-to-date list of supported browsers and known issues, please refer to the PureWeb STK 4.0 release notes.

Setting the Environment Variables

The PureWeb server requires the `JAVA_HOME` environment variable to be pointing to the correct instance of the JDK. The `Path` environment variable must also be set. On a Windows operating system, to access the environment variables:

1. Press the `Windows` key to bring up the **Start** menu (or **Start** screen in Windows 8).
2. Type “environment variables” to search that string.
3. Select the option **Edit the System Environment Variables** from the search results. This will display the System Properties dialog box.
4. In the Advanced tab, click the **Environment Variables** button.

JAVA_HOME

Check that you have a `JAVA_HOME` entry under System variables:

- If the entry is there and the path points to version 1.7 of the JDK, skip this section and check your `Path` variable, as described in next section.
- If the entry is there, but the path points to the wrong version of the JDK: click the **Edit** button, change the path in the **Variable value** field and click **OK**.
- If the entry is missing:
  a. Click the **New** button.
  b. Type `JAVA_HOME` in the **Variable name** field.
  c. Enter the path to JDK 1.7 in the **Variable value** field.
  d. Click the **OK** button to save your changes.

Path

If your `Path` system variable is missing either the path to the JDK or to Ant:

1. In the list of system variables, select the `Path` variable.
2. Click the **Edit...** button.
3. In the **Variable value** string, add the path to `JAVA_HOME`, if missing: `%JAVA_HOME%\bin;`
4. In the **Variable value** string, add the path to `ANT_HOME`, if missing: `%ANT_HOME%\bin;`
5. Click the **OK** button to save your changes.
6. Click **OK** to close the Environment Variables dialog box, then click **OK** to close the System Properties dialog box.

Note: After changing environment variables, if you had a console window open, you will need to restart it for these changes to take effect.

Installing PureWeb

If you already have PureWeb installed, uninstall it first.

To install the PureWeb STK in a Windows operating system:

1. Double-click on its executable file, `PureWeb_Win_Setup-[version].exe`, and follow the instructions in the wizard. The installation directory path cannot contain any spaces.
2. Install your PureWeb server license, a .lic file which you will have received from Calgary Scientific separately from the STK:
   a. Navigate to `[Installed_directory]\Server\conf\`.
   b. Place the .lic license file into that folder.
   c. Double-click on the **Start PureWeb** desktop icon to start the server.
3. Check that the following environment variables are set correctly:
   - `PUREWEB_HOME:`
     `[Installed_directory]\Server`
   - `PUREWEB_LIBS:`
     `[Installed_directory]\SDK\Redistributable\Libs`

Deploying the Sample Applications

The procedure below builds both the Scribble and Asteroids HTML5 clients.

Note: You should build the service first, as described in the `Quick Start Guide` for the service, before following the procedure below.

Service applications are available in C#, C++ and Java for Scribble and Java for Asteroids.

If you do not have Microsoft Visual Studio installed, it is usually easier to get started by installing Java and building the Java service application.

1. Navigate to one of the following directories, depending on which client you want to work with:
   - For Scribble:
     `[Installed_directory]\SDK\Samples\Scribble\ScribbleClient HTML5`
   - For Asteroids:
     `[Installed_directory]\SDK\Samples\Asteroids\AsteroidsClie ntHTML5`
2. Double-click on the `deploy.bat` file.

You should now see a either a `ScribbleApp.html` or an `AsteroidseApp.html` file in the following folder:

- `[Installed_directory]\Server\webapp`

Chapter

3 Applications Overview

This chapter contains instructions on how to run the Scribble and Asteroids sample applications, and describes the functionality available in each.

Note: The default server and port for the PureWeb server is `localhost:8080`. Replace as appropriate if this is not correct for your implementation of the server.

Launching the Applications

Follow the procedure below to launch Scribble or Asteroids:

1. If the server is not running, launch it using the **Start PureWeb** desktop icon.
2. Open an HTML5-compliant browser. We recommend using one of the browsers listed in the PureWeb STK 4.0 Release Notes.
3. Navigate to the relevant client test page URL:
   - localhost:8080/ScribbleTestPage.html
   - localhost:8080/AsteroidsTestPage.html

   In both cases, you will have the option to launch the HTML5 demo client with or without diagnostics. For more information about the option with diagnostics, see “Diagnostics Panel” on page 13.
4. A login screen will appear. Enter `admin` into both the **Name** and **Password** fields and click the **Sign In** button.

The client application will start in the browser, and the service application will be started automatically on the PureWeb server.

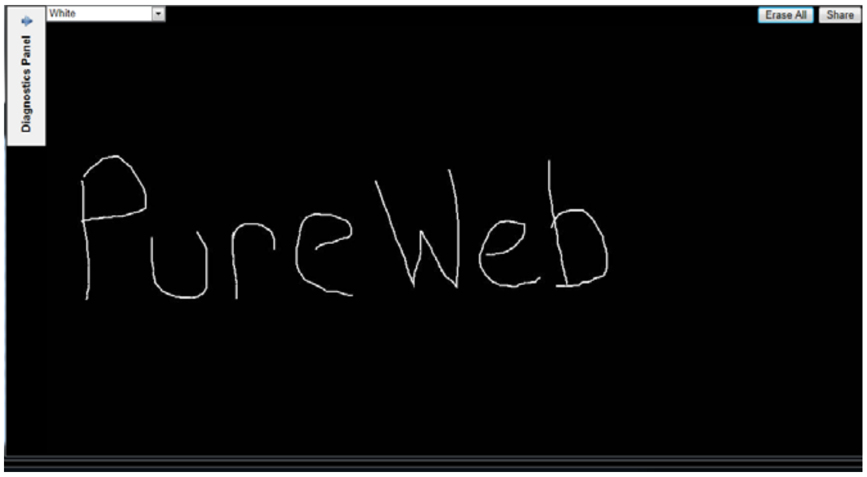
Exercising the Functionality

This section briefly describes the functionality available in each of the two sample applications.

Scribble

Scribble is a simple canvas that users can use to draw lines.

When you first launch the application, it displays a blank canvas. You can draw lines on the canvas by left-clicking the mouse and dragging.



Scribble also allows users to:

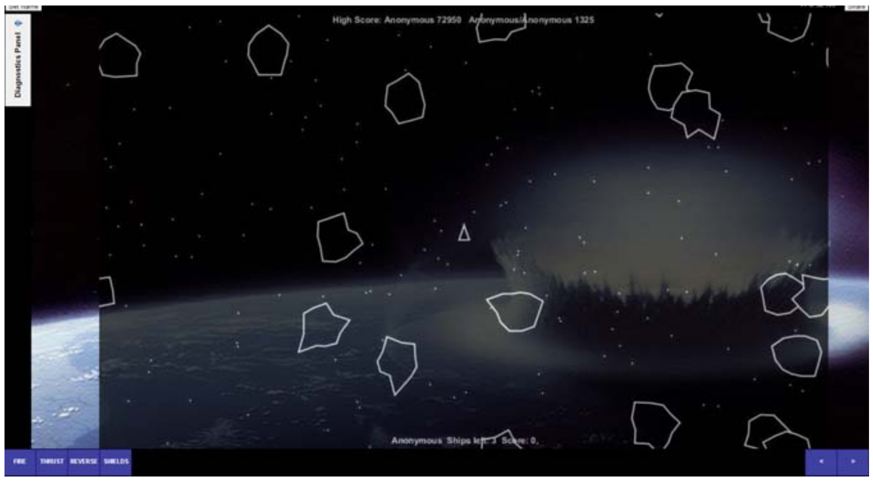
- change colors by selecting a color from the drop-down list,
- clear the canvas by clicking on the **Erase All** button,
- share the application with other users by clicking the **Share** button (see Sharing the Application on page 14),
- display the Diagnostics Panel by clicking on the small arrow button located in the upper-left corner of the client window (if you launched the application with the `_diagnostics=true` parameter); see the section “Diagnostics Panel” on page 13 for more information.

Notice how the service and client windows are linked:

- When you draw in the browser window, you are actually sending commands to draw on the service application, and what is drawn there is mirrored in the browser's view.
- When you resize the client browser window, the service window also resizes (works with the C# service only)
- When you close the client window or navigate away from the page, the service window also closes.

Asteroids

The Asteroids sample application is a PureWeb-enabled implementation of the classic video game with a twist -- it supports both single-player and two-player modes; each player independently controls his or her own ship.


High Score: Anonymous 72950 Anonymous/Anonymous 1325
Diagnostics Panel
Anonymous Ships left: 3 Score: 0
FIRE
THRUST
REVERSE
SHIELDS

Single-Player Mode

- Press any key to start the game.
- Use the up/down arrow keys to move forward/reverse.
- Use the left/right arrow keys to change your heading.
- Press the space bar to fire torpedoes.
- Activate shields by pressing the `s` key. Each player gets 5 seconds of shield time per game.
- Collide with an asteroid with your shields up to destroy the asteroid and collect the points.
- Monitor the number of frames per second in the FPS field.
- Display the Diagnostics Panel by clicking on the small arrow button located in the upper-left corner of the client window (if you launched the application with the diagnostics option enabled); see the section “Diagnostics Panel” on page 13 for more information.
- Share the application with other users by clicking the **Share** button (see Sharing the Application on page 14).

Two-Player Mode

Instructions on how to share the application with another user are provided in section “Sharing an Application” on page 14.

Once the application is shared, additional two-player mode functionality includes:

- Collide with an unshielded opponent with your shields up to destroy their ship and collect all of their points.
- Collide with a shielded opponent with your shields up to create an elastic collision.

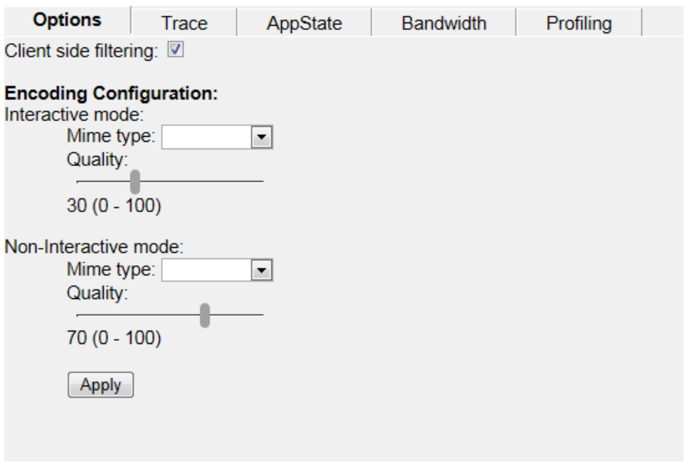
The buttons at the bottom of the screen allow users to play the game on a touch-screen device.

Diagnostics Panel

The Diagnostics Panel debugging tool is described in detail in the Debugging chapter of the *PureWeb STK Developer Guide*, however below is a brief overview of the functionality it offers.

To display the panel (if you launched the application with the diagnostics option enabled), click on the small arrow button located in the upper-left corner of the client window.

The panel is composed of five tabs, as illustrated below.



Client side filtering: ☑

Encoding Configuration:
Interactive mode:
Mime type:
Quality:
30 (0 - 100)

Non-Interactive mode:
Mime type:
Quality:
70 (0 - 100)

- The **Options** tab is used to define options that impact image bandwidth and quality.
- The **Trace** tab is used to examine the messages to and from the server.
- The **AppState** tab displays an XML representation of the application’s state.
- The **Bandwidth** tab is used to measure latency and bandwidth.
- The **Profiling** tab is used to measure performance, such as number of frames per second.

Sharing an Application

Both the Scribble and Asteroids sample applications contain functionality to illustrate collaboration. This section describes how one user can share the application with a second user so that they are both logged in the same session.

A few important points to note before getting started:

- By default, the sample client connects using the hostname of the local host. If you are trying to share the application from another computer, you must modify the URL, `http://localhost:8080/pureweb/...`, by replacing `localhost` with the server's IP address or DNS name.
- Cookie sharing between browser tabs may cause issues with the functionality of the sample application. We recommend that you use two separate browser instances if you want to open two sessions of the same application. Chrome users can also use an incognito window.

First user:

1. Click the **Share** button located in the upper-right corner of the client application.
2. Copy the URL displayed in the popup window and close the window.
3. Provide the URL you copied to the person you want to share the session with.

Second user:

1. Launch the client, using the URL shared by the first user.

Chapter

4 Exploring the Code

One of the main purposes of including sample applications with the STK is for developers who are new to PureWeb to use these samples as a starting point for coding and to experiment with them.

Don't be shy about editing the code and trying new things. That's the best way to learn, and you'll see how quickly you'll become proficient with PureWeb.

This chapter gives some pointers about the code in the sample HTML5 clients; however, it assumes that you are familiar PureWeb concepts such as views, state, commands and event handlers, which are described in the *Developer's Guide*.

For an overview of the service-side code, please refer to the *Quick Start Guide* available for each supported service programming language.

You'll find the files described in this chapter at the following locations:

```
[Installed_directory]\SDK\Samples\Scribble\ScribbleClientHTML5
[Installed_directory]\SDK\Samples\Asteroids\AsteroidsClientHTML5
```

Scribble Client Code Pointers

The `ScribbleApp.html` file sets up the presentation structure for the client.

Page structure

`ScribbleApp.html` is a standard, straight-forward HTML5 page.

Note the `<!DOCTYPE html>` tag at the very start of the page; it triggers HTML standards mode in the page, and is necessary for HTML5 documents.

When the page loads, it references some other required files in the header:

- `scribbleapp.css` is a cascading stylesheet file, which describes presentation and positioning rules for the elements on the page.
- `pureweb.min.js` is the packaged up client STK library which is provided when PureWeb is installed.
- `ScribbleApp.js` is the JavasScript code file which contains client-specific code. This is the file you would modify or replace when writing your own client application.

The body of the page contains some standard `div` tags to define regions for the controls along the top of the page, and the PureWeb view.

Page Elements Specific to PureWeb

This section describes in more details some of the elements in the `ScribbleApp.js` file.

View Controls

The `div` tags with the following id and class attributes are PureWeb view controls:

```
<div id="ScribbleView" class="purewebview"></div>
```

There may be more than one view on a page. All views have the same class name, but each must have a unique id. The id must match a view defined in the service application.

This view in the sample application has no content, and this is fine. The content gets populated with the help of PureWeb API calls when the page has finished loading.

Diagnostics Panel Control

The tag for the PureWeb Diagnostics Panel control looks like this:

```
<div id="pwDiagnosticsPanel"></div>
```

The id must be `pwDiagnosticsPanel`. For details see “Enabling the Diagnostics Panel” on page 18.

Other controls

The controls below are standard HTML form controls. They trigger the functions defined in the `ScribbleApp.js` code file. They do not have to be contained within a FORM tag.

```
<button onclick="clearCanvas();">Erase All</button>
<button onclick="generateShareUrl();">Share</button>>
<select onChange="changeScribbleColor();" id="color"></select>
```

startScribble()

Once the page is loaded, the browser runs the code in the `onLoad` event of the page's body tag.

The `startScribble()` function, defined in the `ScribbleApp.js` file, performs the following tasks; do not hesitate to dive in and review the code yourself:

- Adds an event handler for `TOUCH_EVENT_RECEIVED` so that Scribble can handle custom touch events (in this case, a double tap). See “Custom Touch Events” on page 18.

- Initializes the view by creating a new `pureweb.client.View` instance and attaching it to the `ScribbleView` element.
- Builds the URI of the application service, which is in the form:
  `http://<host>:<port>/pureweb/app?name=ScribbleApp`
- Sets up a listener for the `CONNECTED_CHANGED` event.
- Sets up a window handler function to disconnect when the page is unloaded or closed.
- Sets up the list of colors.
- Adds a color change listener to the application's `StateManager`.
- Connects to the service application.

onConnectedChanged()

When the service reports a successful connection to the client, the service application has been started and is ready to receive commands.

- The `CONNECTED_CHANGED` event fires when the state of the connection changes, whether on connect or disconnect. This implies that the code needs to check that the connection has been established.
- Start listening for `STALLED_CHANGED` and `SESSION_STATE_CHANGED` events, to inform the user if connectivity issues are encountered.
- At this point, if `diagnosticsPanel` is present in the page, it is initialized.

Interacting with the PureWeb Service Application

By design, the PureWeb view automatically listens for input events (mouse, keyboard, touch), and passes them through to the service application.

Therefore, developers do not need to do anything special to capture basic user input when clicking and dragging on the canvas.

If you are interested in implementing more advanced touch-based gestures available to mobile browsers, PureWeb provides the ability to override the default touch behavior in the browser, allowing you to implement your own. See “Custom Touch Events” on page 18.

In addition to basic input events, it is possible to send additional commands to the service application.

To trigger an action in the PureWeb service application, two methods are available:

- Send a command to the service application
- Modify the application's state

Sending a command

A simple example of sending a command is in the `clearCanvas()` function.

Note that the command must be available in the service application. This is described in detail in the *PureWeb Developer Guide*.

Modifying Application State

PureWeb maintains the state of the application and keeps it synchronized between the client and the service application.

The client application can write a change to the application state, which is then sent to the service.

This would happen, for example, when the user selects a new drawing color. In this case, the application state change is written to the `ScribbleColor` attribute.

When the service application receives this update, it sets the attribute and redraws the canvas using the new color. The service application redraws itself and sends the updated view to the client.

Enabling the Diagnostics Panel

Earlier we saw how we can request diagnostics in the URL, using `&_diagnostics=true` parameter

The client application must include and initialize the Diagnostics Panel control for this to work.

The control is inserted into the page, simply by adding:

```
<div id="pwDiagnosticsPanel"></div>
```

When the control is present, it is initialized by calling a PureWeb API function from the `startScribble()` function inside the `ScribbleApp.js` file:

```
pureweb.client.diagnostics.initialize();
```

Custom Touch Events

The Scribble sample application contains code which demonstrates the PureWeb functionality that allows developers to override default touch behaviors in client applications.

Overriding gestures on mobile devices is not mandatory; it is an entirely optional feature that allows common gestures such as pinching and swiping to trigger specific actions within your PureWeb enabled client.

startScribble()

The `startScribble()` method adds a listener for the `TOUCH_EVENT_RECEIVED` event. The handler for this checks if the touch event includes more than one point; if it does, it then sets `cancelBubble` to `true`, which prevents the event from bubbling further up the handler hierarchy.

doubleTap()

The `doubleTap()` method, which can be found at the bottom of `Scribble.js`, listens for double-tap gesture interactions with the PureWeb view. This method is a modification of the JQuery double-tap method. Although the sample application uses JQuery, there are several other JavaScript libraries that handle touch events and gestures, and you are free to use the library of your choice when you develop your own PureWeb applications.

In the Scribble sample, the event handler for `doubleTap()` sends a PureWeb `clear` command to the service application, which then clears the screen. This is of course application-specific behavior, and what happens when a specific touch gesture is captured will vary for each application.

Asteroids Client Code Pointers

The `AsteroidsApp.html` file sets up the presentation structure for the PureWeb client.

Its page structure and definition for views and the Diagnostics Panel is very similar to that of the Scribble application described earlier in this chapter.

Page Elements Specific to PureWeb

The section below focuses on the code aspects which are different between Scribble and Asteroids.

Collaboration

The following are buttons for collaboration in two-player mode; they are standard HTML form controls which trigger functions that are defined in the `AsteroidsApp.js` code file.

```
<button id="setNameButton" onclick="getName();">Set
Name</button>
<button id="shareButton"
onclick="generateShareUrl();">Share</button>
```

Frame Per Second Count

The following control is updated every time the view update is received from the service application:

```
<div id="fps-counter" class="fps-counter">FPS:</div>
```

User Input Events

Note the button panel underneath the Asteroids view.

These define which hot areas on the page are enabled for mouse and touch events.

When a user clicks or touches one of these areas, the client application simulates a keypress event and sends it to the service application.

This is a powerful feature of PureWeb: by overlaying touch-enabled controls, traditional applications can be modernized for new touchscreen devices.

startAsteroids()

This method performs the same steps as `startScribble()`, with one addition: it contains a handler function whose purpose is to respond when the `Level` attribute changes in the session state.

onConnectedChanged()

When the service reports a successful connection to the client, this means this service has been started and is ready to receive commands.

- The `CONNECTED_CHANGED` event fires when the state of the connection changes, whether on connect or disconnect. This implies that the code needs to check that the connection has been established.
- Start listening for `STALLED_CHANGED` and `SESSION_STATE_CHANGED` events, to inform the user if connectivity issues are encountered.
- If there are no connectivity issues, the view can be initialized creating a new `pureweb.client.View` instance and attaching it to the `Asteroids` div element.
- At this point, if `diagnosticsPanel` is present in the page, it is initialized.
- Start the frames per second (FPS) counter

Chapter

5 Debugging

Using the sample application as models, this chapter describes how you would debug a PureWeb HTML5 client in a web browser.

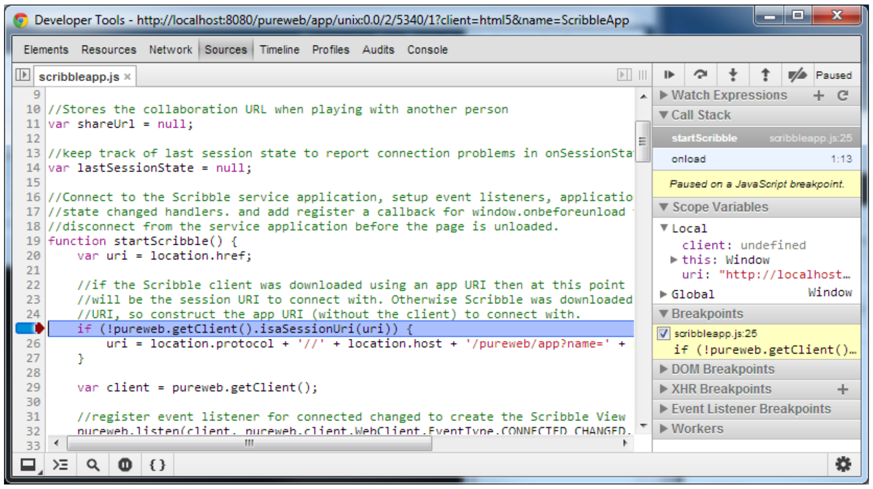
Note: The instructions in this section assume that you are working in a Chrome browser and not using any third-party libraries.

If you are using a different browser or different libraries, please adapt accordingly.

Other sources of debugging information include the Diagnostics Panel section of the Debugging chapter in the *Developer's Guide*, as well as the Debugging chapter in each of the Quick Start Guides for service development.

1. Open the client application in the browser.
2. Open the browser's Developer Tools (select **Tools > Developer Tools** from the main menu.)
3. Choose the **Sources** tab
4. Locate the script file for the application you are debugging, for example `ScribbleApp.js` or `AsteroidsApp.js`. In Chrome, you do this by clicking the small boxed arrow in the top left, then navigating to the file in the list.
5. Add a breakpoint by clicking in the left margin on a line in the `startScribble()` or `startAsteroids()` function.
6. Refresh the page in the browser. The debugger should pause execution at the line where you set the breakpoint.

Once in a breakpoint, you can step through the code, observe values of variables, etc.



```

//Stores the collaboration URL when playing with another person
var shareUrl = null;

//keep track of last session state to report connection problems in onSessionSta
var lastSessionState = null;

//Connect to the Scribble service application, setup event listeners, applicatio
//state changed handlers. and add register a callback for window.onbeforeunload
//disconnect from the service application before the page is unloaded.
function startScribble() {
    var uri = location.href;

    //if the Scribble client was downloaded using an app URI then at this point
    //will be the session URI to connect with. Otherwise Scribble was downloaded
    //URI, so construct the app URI (without the client) to connect with.
    if (!pureweb.getClient().isSessionUri(uri)) {
        uri = location.protocol + '//' + location.host + '/pureweb/app?name=' +
    }

    var client = pureweb.getClient();

    //register event listener for connected changed to create the Scribble View
    pureweb.listen(client, pureweb.client.WebClient.EventType.CONNECTED_CHANGED,
```

```
startScribble    scribbleapp.js:25
onload           1:13
Paused on a JavaScript breakpoint.
Local
  client: undefined
  this: Window
  uri: "http://localhost…
Global           Window
scribbleapp.js:25
  if (!pureweb.getClient()…
```