Better Web Development with WebKit Remote Debugging

Ashutosh Jagdish Sharma | Senior Computer Scientist | Adobe

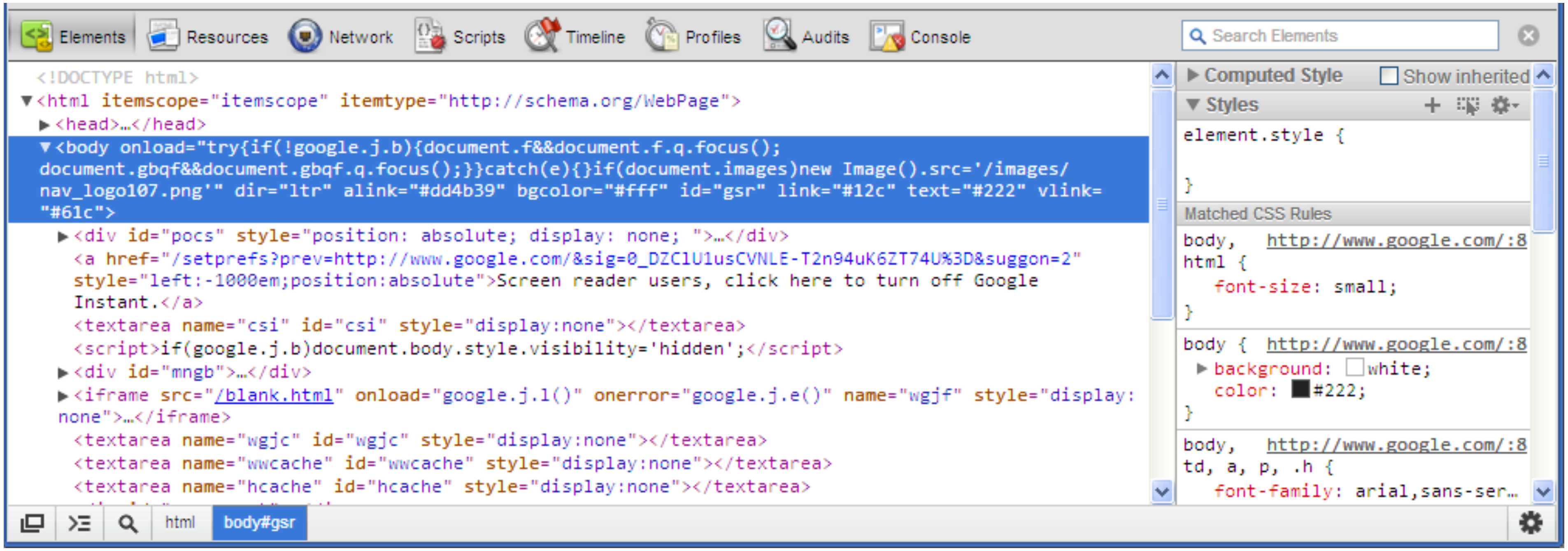
Developer Track | WWW 2012 | Lyon

Agenda

- WebKit Remote Debugging Protocol
- Demos and Code Walkthroughs
  - Catching uncaught JavaScript exceptions
  - Inspecting the computed style for a visually selected node

Introduction - WebKit Remote Debugging Protocol

- JSON-RPC 2.0
- Supported by Chrome/Chromium, Chrome for Android, RIM Playbook
- Current version: 1.0 (April 9, 2012)
- Used by the Web Inspector front-end (Chrome Developer Tools)



```
<!DOCTYPE html>
▼<html itemscope="itemscope" itemtype="http://schema.org/WebPage">
 ▶<head>…</head>
 ▼<body onload="try{if(!google.j.b){document.f&&document.f.q.focus();
 document.gbqf&&document.gbqf.q.focus();}}catch(e){}if(document.images)new Image().src='/images/
 nav_logo107.png'" dir="ltr" alink="#dd4b39" bgcolor="#fff" id="gsr" link="#12c" text="#222" vlink=
 "#61c">
   ▶<div id="pocs" style="position: absolute; display: none; ">…</div>
    <a href="/setprefs?prev=http://www.google.com/&sig=0_DZC1U1usCVNLE-T2n94uK6ZT74U%3D&suggon=2"
    style="left:-1000em;position:absolute">Screen reader users, click here to turn off Google
    Instant.</a>
    <textarea name="csi" id="csi" style="display:none"></textarea>
    <script>if(google.j.b)document.body.style.visibility='hidden';</script>
   ▶<div id="mngb">…</div>
   ▶<iframe src="/blank.html" onload="google.j.l()" onerror="google.j.e()" name="wgjf" style="display:
   none">…</iframe>
    <textarea name="wgjc" id="wgjc" style="display:none"></textarea>
    <textarea name="wwcache" id="wwcache" style="display:none"></textarea>
    <textarea name="hcache" id="hcache" style="display:none"></textarea>
```

```
element.style {

}

body,    http://www.google.com/:8
html {
   font-size: small;
}

body {   http://www.google.com/:8
 ▶ background: white;
   color: #222;
}

body,    http://www.google.com/:8
td, a, p, .h {
   font-family: arial,sans-ser…
```

Why Use the Remote Debugging Protocol?

- Debug over the wire
  - Chrome for Android
  - RIM Playbook
- Enhance existing tools
- Build custom tools
- Integrate with IDEs

- Sample Use cases
  - Tracking profiling data over time
  - Logging and filtering console messages

WebKit Remote Debugging Protocol

- Web browser as a server
- Clients can reside in another process
  - Useful for mobile devices that have a limited screen area
- Asynchronous communication over a websocket
- Inspector.json

API Surface

- Divided into categories (called *domains*)
- Each domain contains:
  - Commands
    - Allow clients to send requests to the browser
    - *e.g. DOM.querySelectorAll* is a command that requests the set of nodes that match a given selector
  - Events
    - Used for asynchronous notifications
    - Used as responses to commands
    - *e.g. DOM.childNodeRemoved* is an event dispatched when a child node is removed from its parent

Domains

- Protocol version 1.0:
  - **Console** - Interaction with the JavaScript console
  - **DOM** - DOM read/write operations
  - **DOMDebugger** - Breakpoints on DOM events and operations
  - **Debugger** - JavaScript debugging capabilities
  - **Network** - Tracking network activities of the page
  - **Page** - Actions and events related to the inspected page
  - **Runtime** - JavaScript runtime
  - **Timeline** - Instrumentation records for the page run-time

Hidden Domains

- ApplicationCache
- CSS
- DOMStorage
- Database
- FileSystem
- IndexedDB
- Inspector
- Memory
- Profiler
- Worker

- No guarantee of backwards compatibility
- Internally used by the Web Inspector
- Visible domains can also have some hidden commands and events

Development Setup

- Chrome or Chromium build
- Launch with remote debugging enabled:

```
{Path to Chromium} --remote-debugging-port=9222
```

- Navigate to http://localhost:9222/ to see a list of inspectable pages

Development Setup

- Navigate to http://localhost:9222/json to see details relevant to remote debugging:

```
{
        "devtoolsFrontendUrl": "/devtools/devtools.html?host=localhost:9222&page=2",
        "faviconUrl": "http://www.google.com/favicon.ico",
        "thumbnailUrl": "/thumb/http://www.google.com/",
        "title": "Google",
        "url": "http://www.google.com/",
        "webSocketDebuggerUrl": "ws://localhost:9222/devtools/page/2"
}
```

- Multiple remote debugging connections not available (Chrome Developer Tools)

Code Walkthroughs

- Chromium Build 127895 is used for the live demos
- Remote debugging client running inside the browser itself
  - to minimize the required setup
- Navigate to http://localhost:9222/json and invoke a bookmarklet

```
javascript:(function(){
    function loadScript(scriptURL) {
        var scriptElem = document.createElement("script");
        scriptElem.setAttribute("language", "JavaScript");
        scriptElem.setAttribute("src", scriptURL);
        document.body.appendChild(scriptElem);
    }
    loadScript("{path-to-javascript-file.js}");
})()
```

1. Catching Uncaught JavaScript Exceptions

- Replace {path-to-javascript-file.js} with:

  http://marple.host.adobe.com/webkit/demo/exceptions.js
- Create a new bookmarklet with the modified code
- Invoke the bookmarklet
- Select a target page
- The client connects to the remote debugging server at the URL specified by *webSocketDebuggerUrl*
- Sample page
  - http://marple.host.adobe.com/webkit/demo/exception.html

exception.js – Sending a Command to the Server

```
function process(webSocketDebuggerUrl, pageUrl) {
    var dbg = Debugger.getDebugger(webSocketDebuggerUrl);
    dbg.connect().done(function() {
        dbg.sendCommand("Debugger.enable").done(function() {
            dbg.sendCommand("Debugger.setPauseOnExceptions", { state: "uncaught" } );
        });
    });
}
```

- Debugger
    - Helper class to manage the connection with the remote debugging server
    - Utilizes jQuery's Deferred functionality to manage and chain asynchronous callbacks.
    - Not related to the *Debugger* domain of the remote debugging protocol

Workflow

- Uncaught exception thrown on the target page
- *Debugger.paused* event dispatched
- Received as a message on the debugger websocket in the client
- JSON data packet has a *method* property with the value *Debugger.paused*
- Extract useful information
- Resume the debugger (to avoid halting the page)

exception.js – Obtain Information about the Exception

```
if(json.params.reason === "exception") {
    var errorName = json.params.data.className;
    var callFrames = json.params.callFrames;
    var callStack = "";
    for(var i = callFrames.length - 1; i >= 0; i--) {
        if(callStack !== "")
            callStack += " -> ";
        callStack += callFrames[i].functionName + "()";
    }
    alert("Exception: " + errorName + "\n" + "Callstack: " + callStack);
    self.sendCommand("Debugger.resume");
}
```

2. Inspecting the Computed Style for a Visually-Selected Node

- Replace {path-to-javascript-file.js} with:

  http://marple.host.adobe.com/webkit/demo/computedStyle.js
- Create a new bookmarklet with the modified code
- Invoke the bookmarklet:
- Select a target page
- The client connects to the remote debugging server at the URL specified by *webSocketDebuggerUrl*

computedStyle.js

- Visually selecting an element on the target page

```
dbg.sendCommand("DOM.getDocument")
.done(function(response) {
    dbg.sendCommand("Inspector.enable")
    .done(function(response) {
        var config = {
            showInfo: true,
            contentColor: { r: 255, g: 0, b: 0, a: 0.5 },
            paddingColor: { r: 255, g: 204, b: 153, a: 0.5 },
            marginColor: { r: 255, g: 255, b: 204, a: 0.5 }
        };
        dbg.sendCommand("DOM.setInspectModeEnabled", { enabled: true, highlightConfig: config });
    });
});
```

- An *Inspector.Inspect* event is received when the user selects a node

computedStyle.js

- Handling the *Inspector.Inspect* event

```
self.sendCommand("DOM.requestNode", { objectId: objectId })
.done(function(response) {
    var nodeId = response.result.nodeId;
    self.sendCommand("CSS.getComputedStyleForNode", { nodeId: nodeId })
    .done(function(response) {
        var result = response.result.computedStyle;
        var computedStyle = {};
        for(var i = 0; i < result.length; i++) {
            var s = result[i];
            computedStyle[s["name"]] = s["value"];
        }
        alert("margin-bottom: " + computedStyle["margin-bottom"]);
    });
});
```

- CSS and Inspector are hidden domains – their usage is discouraged

Miscellaneous

- Chrome/Chromium also exposes remote debugging to browser extensions via the *chrome.debugger* extension API
- Other browsers
  - Chrome for Android – can be debugged remotely over USB
  - Firefox
    - Remote debugging clients can connect to Firebug (similar to Chrome Developer Tools) via its Crossfire extension
  - Firefox Mobile (Fennec)
    - Remote debugging in the works
    - Some patches allow remote JS debugging

Acknowledgments

- My thanks to
  - The WebKit team that developed the WebKit Remote Debugging Protocol
  - Narciso Jaramillo (@rictus on Twitter) who wrote the *Debugger* helper class that manages the remote debugging connections

Contact

- Email: ashutosh@adobe.com
- Twitter: @zorder

References

- JSON schema for the remote debugging protocol
  - http://trac.webkit.org/browser/trunk/Source/WebCore/inspector/Inspector.json
- Chrome Developer Tools: Remote Debugging
  - https://developers.google.com/chrome-developer-tools/docs/remote-debugging
- Remote Debugging Protocol 1.0
  - https://developers.google.com/chrome-developer-tools/docs/protocol/1.0/
- Chrome's *Dev Channel builds*
  - http://www.chromium.org/getting-involved/dev-channel
- Nightly Chromium builds
  - http://commondatastorage.googleapis.com/chromium-browser-continuous/index.html
- Chrome Canary builds
  - http://tools.google.com/dlpage/chromesxs

References

- WebKit Remote Debugging (WebKit Blog)
  - http://www.webkit.org/blog/1620/webkit-remote-debugging/
- Bookmarklet
  - http://en.wikipedia.org/wiki/Bookmarklet
- Chromium Build 127895 used for the live demos
  - http://commondatastorage.googleapis.com/chromium-browser-continuous/Mac/127895/chrome-mac.zip
  - http://commondatastorage.googleapis.com/chromium-browser-continuous/Win/127895/chrome-win32.zip
  - http://commondatastorage.googleapis.com/chromium-browser-continuous/Linux/127895/chrome-linux.zip
- jQuery's Deferred Object
  - http://api.jquery.com/category/deferred-object/
- Source code for the tool to alert the user on uncaught JavaScript exceptions
  - http://marple.host.adobe.com/webkit/demo/exceptions.js

References

- Sample web page which has code that throws an uncaught exception
  - http://marple.host.adobe.com/webkit/demo/exception.html
- Source code for the tool to inspect the computed style of a visually-selected node
  - http://marple.host.adobe.com/webkit/demo/computedStyle.js
- Chrome extension API to expose the remote debugging protocol to browser extensions
  - http://code.google.com/chrome/extensions/debugger.html
- Firebug
  - http://getfirebug.com/whatisfirebug
- Crossfire
  - http://getfirebug.com/wiki/index.php/Crossfire
- Remote debugging in Firefox Mobile
  - http://lucasr.org/2012/03/28/remote-debugging-in-firefox-mobile/