



Troubleshooting, Debugging, and Cross-Browser Issues

Bok, Jong Soon
jongsoon.bok@gmail.com
www.javaexpert.co.kr

Simple Ways to Debug

- Is to use some form of output functionality to check the values of variables after processing.

i.e. `alert(someVariable);`

`document.write(someVariable);`

- If interested in a quick check, don't want to open a debugger, add the `alert` box, check the values you need to check, and then immediately remove the alert box.

Simple Ways to Debug (Cont.)

```
<script type="text/javascript">
function sumNumbers(numArray) {
    var result = 0;
    if (numArray.length > 0) {
        for (var i = 0; i < numArray.length; i++) {
            if (typeof numArray[i] == "number") {
                result += numArray[i];
            } else {
                result = NaN;
                break;
            }
        }
    } else {
        result = NaN;
    }
    return result;
}
var ary = new Array(1,15,"three",5,5);
var res = sumNumbers(ary); // res is NaN
if (isNaN(res)) alert("Encountered a bad array or array element");
</script>
```

Development and Debugging Tools by Browser

- Google Chrome Developer Tools

- <https://developers.google.com/chrome-developer-tools/>

- Firefox Firebug

- <http://getfirebug.com/faq/>

- IE Built-in Debugger

- [http://msdn.microsoft.com/en-us/library/ie/gg699336\(v=vs.85\).aspx](http://msdn.microsoft.com/en-us/library/ie/gg699336(v=vs.85).aspx)

- Opera Dragonfly

- <http://www.opera.com/dragonfly/>

- Safari Web Development Tools

- <https://developer.apple.com/technologies/safari/developer-tools.html>

Sample Code to Debugging

```
2 <html>
3 <head>
4 <title> JavaScript Debugging Example </title>
5 <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
6 <script type="text/javascript">
7     var display;
8     function init() {
9         display = document.getElementById("results");
10    }
11    function firstParam() {
12        //set breakpoint here
13        var a = 5;
14        secondParam(a);
15    }
16    function secondParam(a) {
17        var b = 10;
18        thirdParam(a, b);
19    }
20    function thirdParam(a, b) {
21        var c = 15;
22        var d = a + b + c;
23        if (window.console && window.console.log) {
24            window.console.log(a + " + " + b + " + " + c + " = " + d);
25        } else {
26            display.innerHTML = a + " + " + b + " + " + c + " = " + d;
27        }
28    }
29 </script>
30 </head>
31 <body onload="init()">
32     <p><button onclick="firstParam();">Run</button></p>
33     <div id="results"></div>
34 </body>
35 </html>
```

Sample Code to Debugging (Cont.)

```
2 <html>
3 <head>
4   <title> New Document </title>
5   <meta http-equiv="Content-Type" content="text/html; charset=utf-8" >
6   <script type="text/javascript" >
7     function hello(){
8       var msg = "Hello, World!";
9       console.log("%s", msg);
10    }
11  </script>
12 </head>
13 <body onload="hello()" >
14   <p>Hi</p>
15 </body>
16 </html>
```

Chrome Developer Tools

- Select the **Wrench menu**  or  at the top-right of your browser window, then select **Tools -> Developer tools**.
- **Right-click** on any page element and select **Inspect element**.



Chrome Developer Tools (Cont.)

- Resources

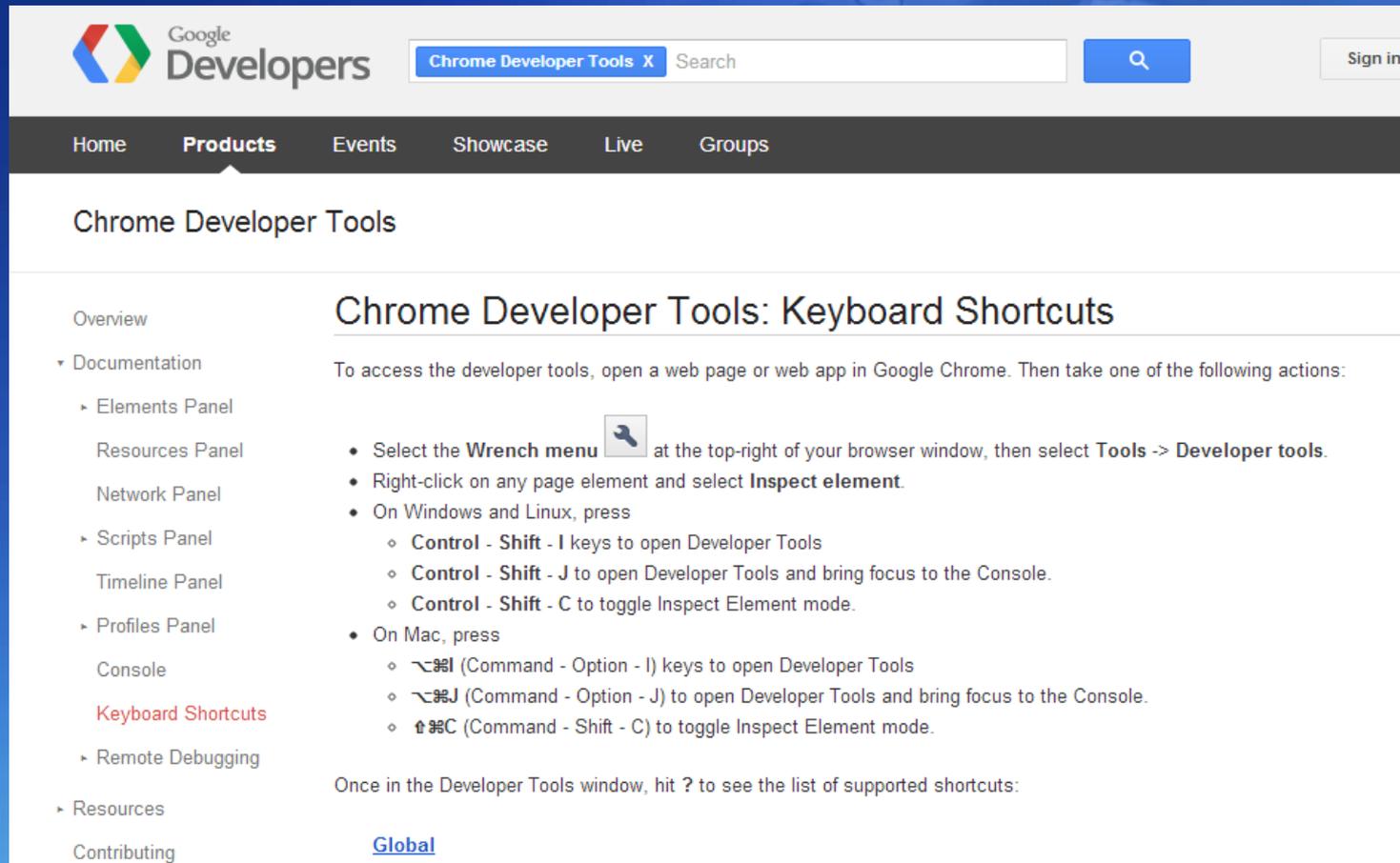
- <http://www.youtube.com/watch?v=n0Ew9iiopwI>
- <http://www.youtube.com/watch?v=htZAU7FM7GI>

Chrome Developer Tools (Cont.)

- On Windows and Linux, press
 - **Control - Shift - I** keys to open Developer Tools
 - **Control - Shift - J** to open Developer Tools and bring focus to the Console.
 - **Control - Shift - C** to toggle Inspect Element mode.
- On Mac, press
 - **⌘I** (**Command - Option - I**) keys to open Developer Tools
 - **⌘J** (**Command - Option - J**) to open Developer Tools and bring focus to the Console.
 - **⌘C** (**Control - Option - C**) to toggle Inspect Element mode.

Chrome Developer Tools: Keyboard Shortcuts

- <https://developers.google.com/chrome-developer-tools/docs/shortcuts>



The screenshot shows the Google Developers website. At the top left is the Google Developers logo. To its right is a search bar containing the text "Chrome Developer Tools X" and a search icon. Further right is a "Sign in" button. Below the logo and search bar is a navigation menu with links for "Home", "Products", "Events", "Showcase", "Live", and "Groups". The "Products" link is highlighted with a white arrow. Below the navigation menu is the main heading "Chrome Developer Tools". On the left side, there is a sidebar menu with the following items: "Overview", "Documentation" (expanded), "Elements Panel", "Resources Panel", "Network Panel", "Scripts Panel", "Timeline Panel", "Profiles Panel", "Console", "Keyboard Shortcuts" (highlighted in red), "Remote Debugging", "Resources", and "Contributing". The main content area is titled "Chrome Developer Tools: Keyboard Shortcuts". It begins with the text: "To access the developer tools, open a web page or web app in Google Chrome. Then take one of the following actions:". This is followed by a list of instructions, each starting with a bullet point. The first instruction includes a small wrench icon. The instructions are: "Select the Wrench menu

Chrome Developer Tools (Cont.)

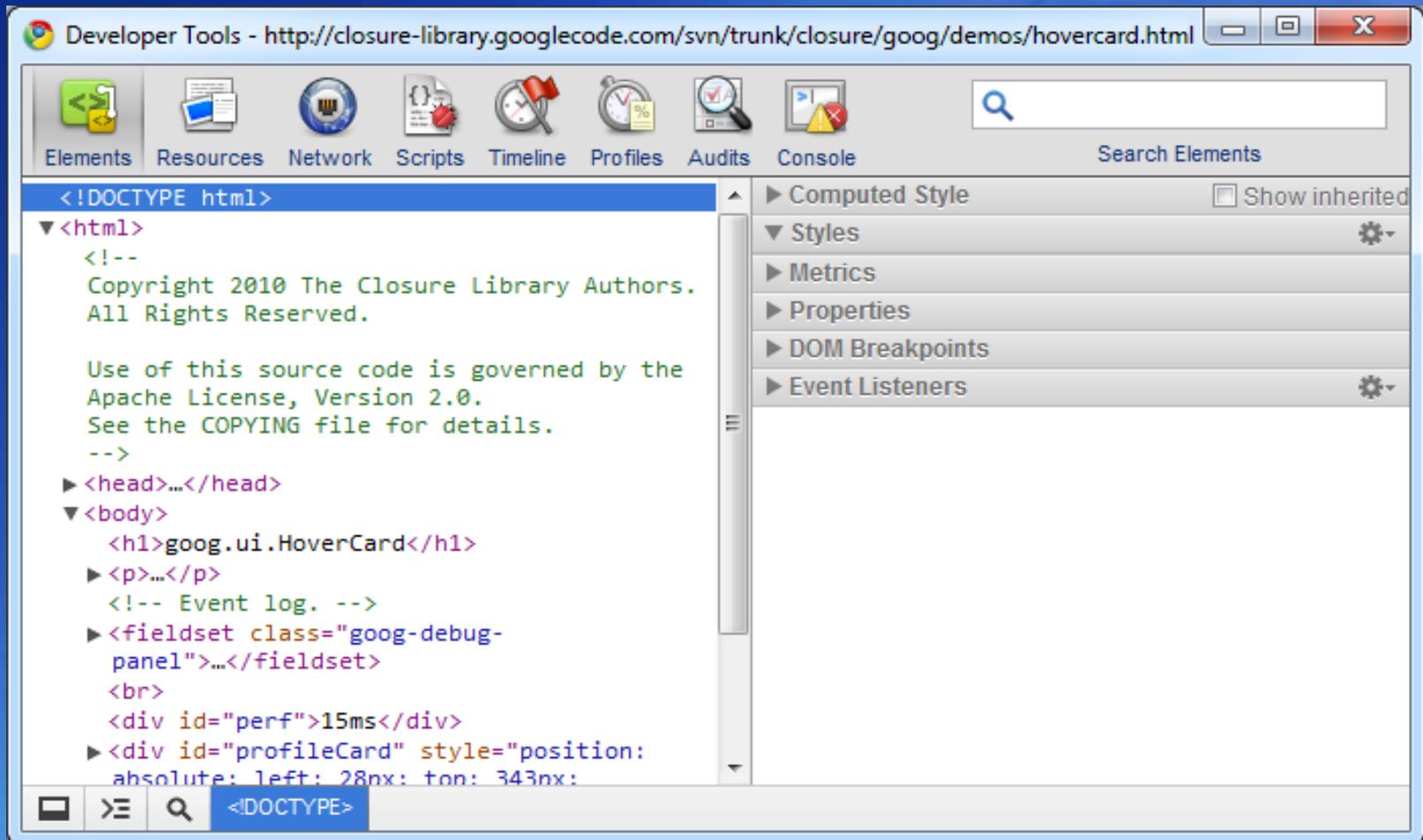
- Are organized into task-oriented groups.
- Are represented by icons in the toolbar at the top of the window.



- Lets you work with a specific type of page or app information, including DOM elements, resources, and scripts.
- Provides a search field that enables you to search the current panel.

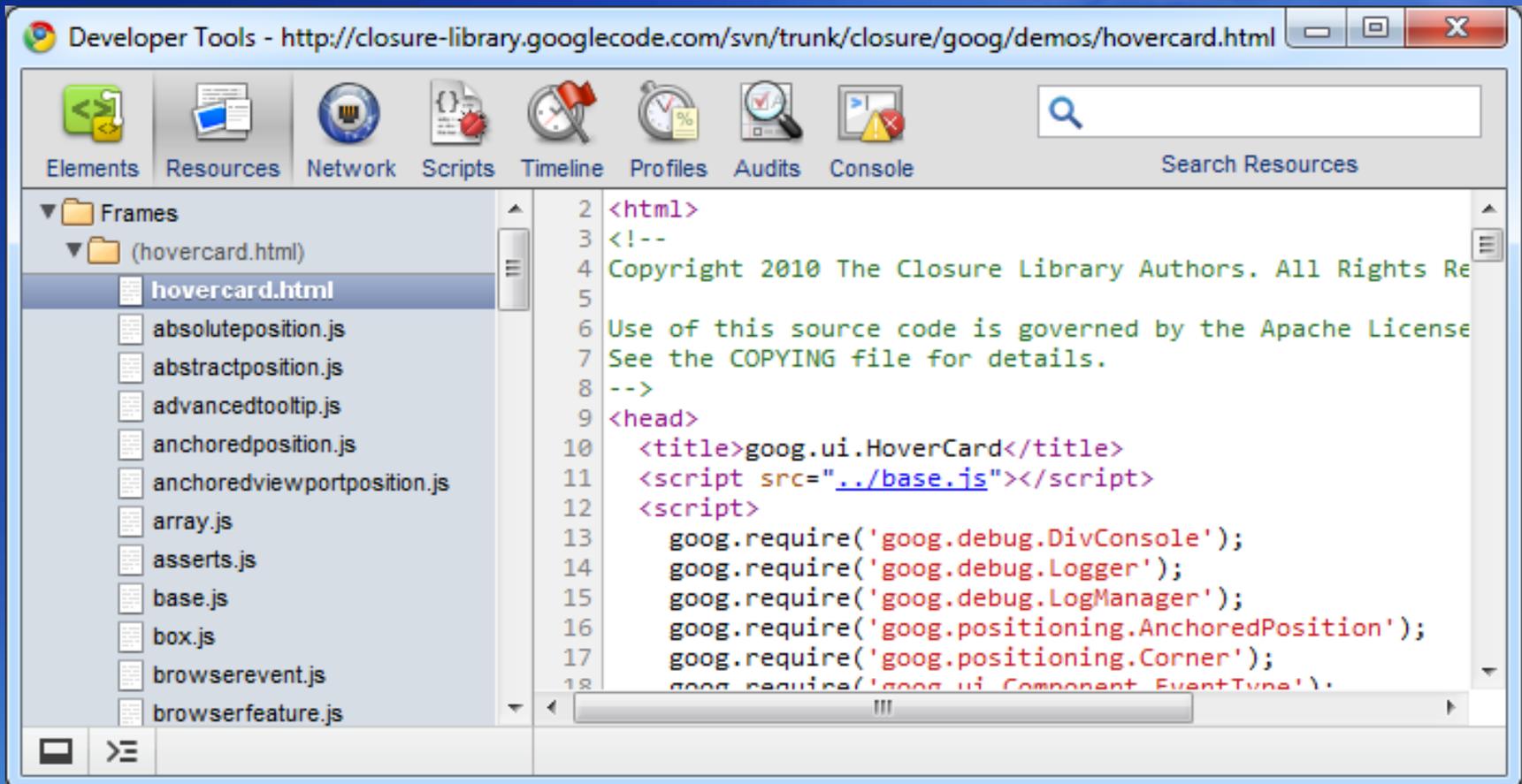
Chrome Developer Tools (Cont.)

• Elements Panel



Chrome Developer Tools (Cont.)

Resources Panel



The screenshot displays the Chrome Developer Tools interface, specifically the Resources Panel. The browser's address bar shows the URL: `http://closure-library.googlecode.com/svn/trunk/closure/goog/demos/hovercard.html`. The Resources Panel toolbar includes icons for Elements, Resources, Network, Scripts, Timeline, Profiles, Audits, and Console, along with a search box labeled "Search Resources".

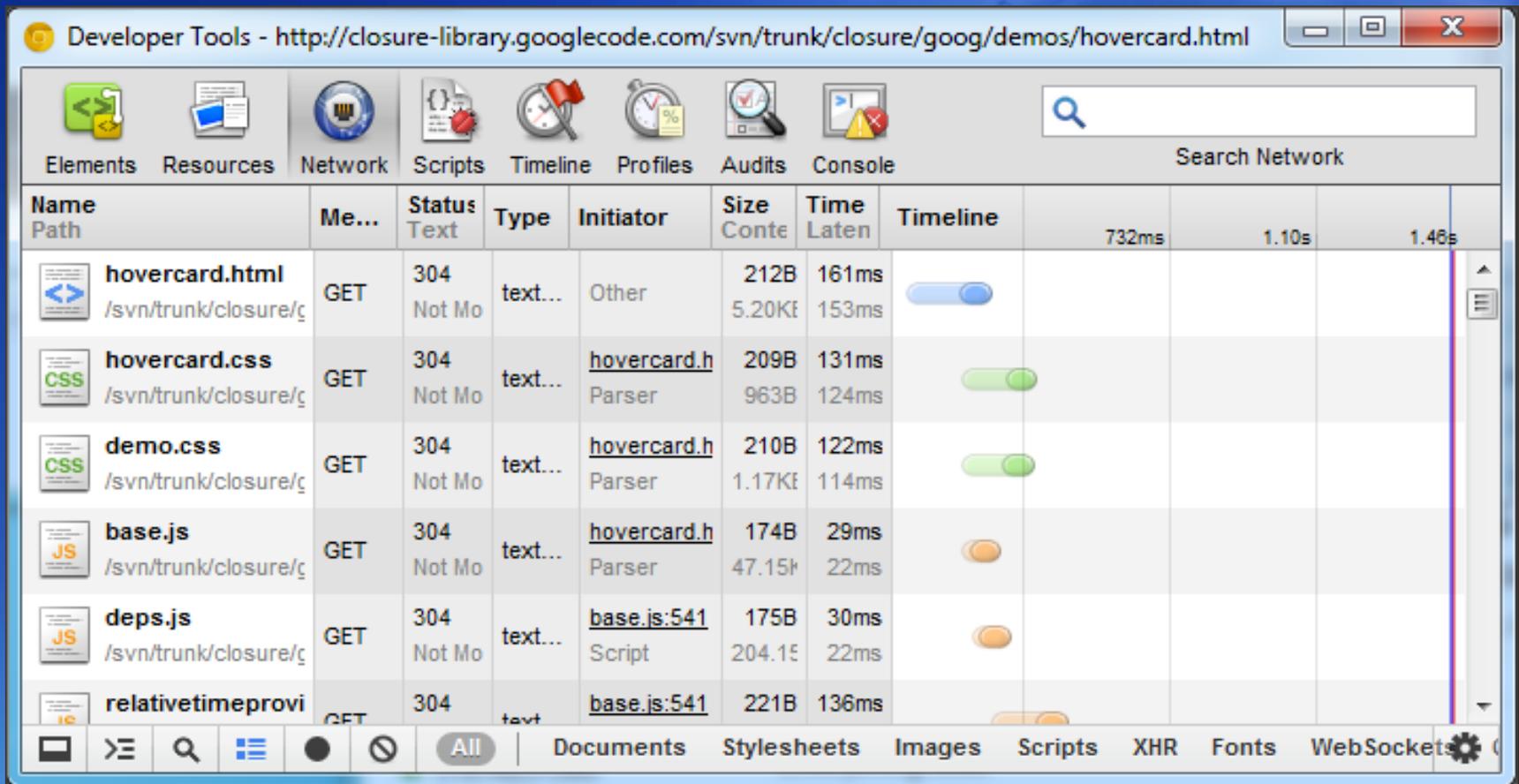
The Resources Panel is expanded to show the file structure of the page. The "Frames" folder is expanded, revealing the "(hovercard.html)" folder. Inside this folder, the "hovercard.html" file is selected. Below it, a list of JavaScript files is shown, including `absoluteposition.js`, `abstractposition.js`, `advancedtooltip.js`, `anchoredposition.js`, `anchoredviewportposition.js`, `array.js`, `asserts.js`, `base.js`, `box.js`, `browserevent.js`, and `browserfeature.js`.

The main content area of the Resources Panel displays the source code of the selected `hovercard.html` file. The code is as follows:

```
2 <html>
3 <!--
4 Copyright 2010 The Closure Library Authors. All Rights Reserved.
5
6 Use of this source code is governed by the Apache License
7 See the COPYING file for details.
8 -->
9 <head>
10 <title>goog.ui HoverCard</title>
11 <script src="..base.js"></script>
12 <script>
13   goog.require('goog.debug.DivConsole');
14   goog.require('goog.debug.Logger');
15   goog.require('goog.debug.LogManager');
16   goog.require('goog.positioning.AnchoredPosition');
17   goog.require('goog.positioning.Corner');
18   goog.require('goog.ui.Component.EventType');
```

Chrome Developer Tools (Cont.)

- Network Panel



The screenshot shows the Chrome Developer Tools Network Panel. The browser address bar displays the URL: `http://closure-library.googlecode.com/svn/trunk/closure/goog/demos/hovercard.html`. The Network tab is selected, showing a list of network requests. The table below summarizes the visible requests:

Name Path	Method	Status Text	Type	Initiator	Size Conte	Time Laten	Timeline
 hovercard.html /svn/trunk/closure/...	GET	304 Not Mo	text...	Other	212B 5.20Kf	161ms 153ms	
 hovercard.css /svn/trunk/closure/...	GET	304 Not Mo	text...	<u>hovercard.h</u> Parser	209B 963B	131ms 124ms	
 demo.css /svn/trunk/closure/...	GET	304 Not Mo	text...	<u>hovercard.h</u> Parser	210B 1.17Kf	122ms 114ms	
 base.js /svn/trunk/closure/...	GET	304 Not Mo	text...	<u>hovercard.h</u> Parser	174B 47.15f	29ms 22ms	
 deps.js /svn/trunk/closure/...	GET	304 Not Mo	text...	<u>base.js:541</u> Script	175B 204.1f	30ms 22ms	
 relativetimeprovi /svn/trunk/closure/...	GET	304	text	<u>base.js:541</u>	221B	136ms	

The bottom of the panel shows navigation icons and filter tabs: All, Documents, Stylesheets, Images, Scripts, XHR, Fonts, and WebSockets.

Chrome Developer Tools (Cont.)

Scripts Panel

The screenshot shows the Chrome Developer Tools interface with the Scripts Panel open. The browser address bar shows the URL: `http://closure-library.googlecode.com/svn/trunk/closure/goog/demos/hovercard.html`. The Scripts Panel toolbar includes icons for Elements, Resources, Network, Scripts, Timeline, Profiles, Audits, and Console, along with a search box labeled "Search Scripts".

Annotations with arrows point to the following features:

- Tab navigator:** Points to the "hovercard.js" and "popup.js" tabs.
- Each script opens in a tab:** Points to the "popup.js" tab.
- Pause, resume, step through code:** Points to the execution control buttons (Pause, Step Over, Step Into, Step Out, Stop) in the Scripts Panel toolbar.
- Clear all breakpoints:** Points to the "Clear all breakpoints" button in the top right corner of the Scripts Panel.

The main code editor displays the following JavaScript code:

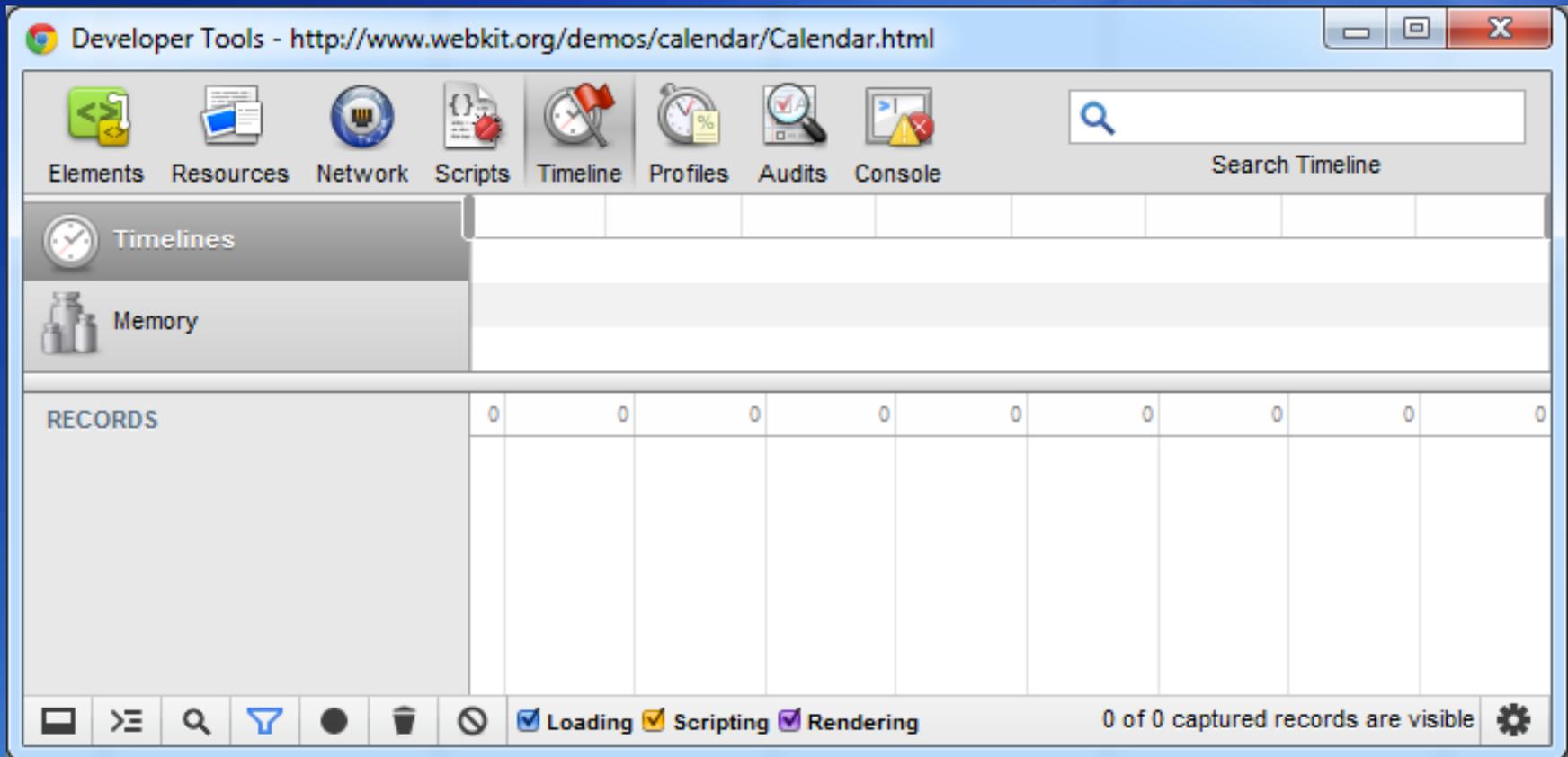
```
1 // Copyright 2006 The Closure Library Authors. All Rights Reserved.
2 //
3 // Licensed under the Apache License, Version 2.0 (the "License");
4 // you may not use this file except in compliance with the License.
5 // You may obtain a copy of the License at
6 //
7 //     http://www.apache.org/licenses/LICENSE-2.0
8 //
9 // Unless required by applicable law or agreed to in writing, software
10 // distributed under the License is distributed on an "AS-IS" BASIS,
11 // WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
12 // implied. See the License for the specific language governing
13 // permissions and limitations under the License.
14
15 /**
16  * @fileoverview Definition of the Popup class.
17  */
```

The right-hand sidebar contains the following panels:

- Watch Expressions: Not Paused
- Call Stack: Not Paused
- Scope Variables: Not Paused
- Breakpoints: No Breakpoints
- DOM Breakpoints
- XHR Breakpoints
- Event Listener Breakpoints
- Workers

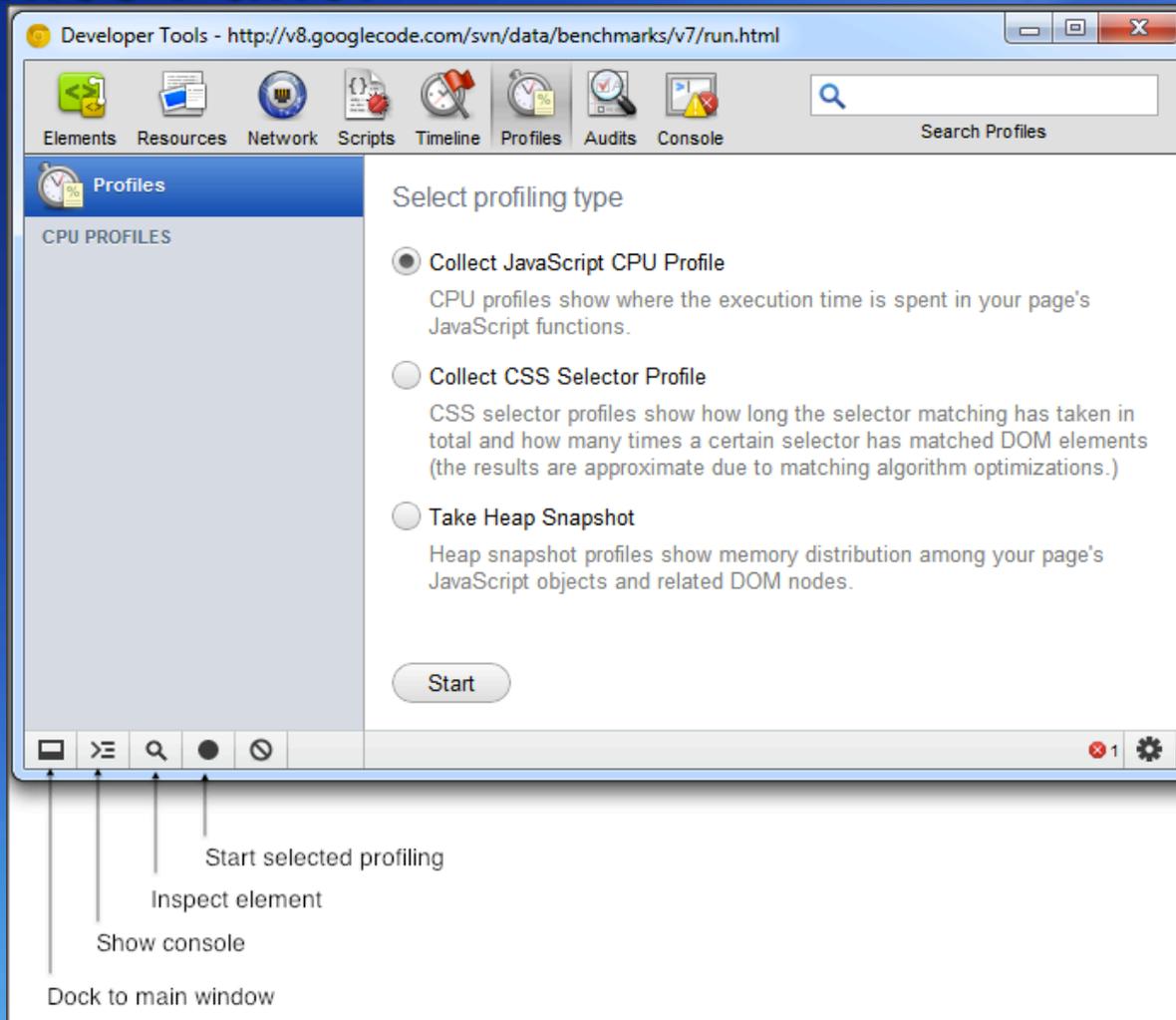
Chrome Developer Tools (Cont.)

- Timeline Panel



Chrome Developer Tools (Cont.)

Profiles Panel



The screenshot shows the Chrome Developer Tools Profiles Panel. The browser window title is "Developer Tools - http://v8.googlecode.com/svn/data/benchmarks/v7/run.html". The top toolbar includes icons for Elements, Resources, Network, Scripts, Timeline, Profiles, Audits, and Console, along with a search bar labeled "Search Profiles". The Profiles panel is active, showing a "CPU PROFILES" section on the left and a "Select profiling type" section on the right. The "Select profiling type" section has three radio button options: "Collect JavaScript CPU Profile" (selected), "Collect CSS Selector Profile", and "Take Heap Snapshot". Each option has a brief description. A "Start" button is located at the bottom of the "Select profiling type" section. The bottom toolbar of the Profiles panel contains icons for "Dock to main window", "Show console", "Inspect element", and "Start selected profiling".

Developer Tools - http://v8.googlecode.com/svn/data/benchmarks/v7/run.html

Elements Resources Network Scripts Timeline Profiles Audits Console Search Profiles

Profiles

CPU PROFILES

Select profiling type

- Collect JavaScript CPU Profile
CPU profiles show where the execution time is spent in your page's JavaScript functions.
- Collect CSS Selector Profile
CSS selector profiles show how long the selector matching has taken in total and how many times a certain selector has matched DOM elements (the results are approximate due to matching algorithm optimizations.)
- Take Heap Snapshot
Heap snapshot profiles show memory distribution among your page's JavaScript objects and related DOM nodes.

Start

Start selected profiling

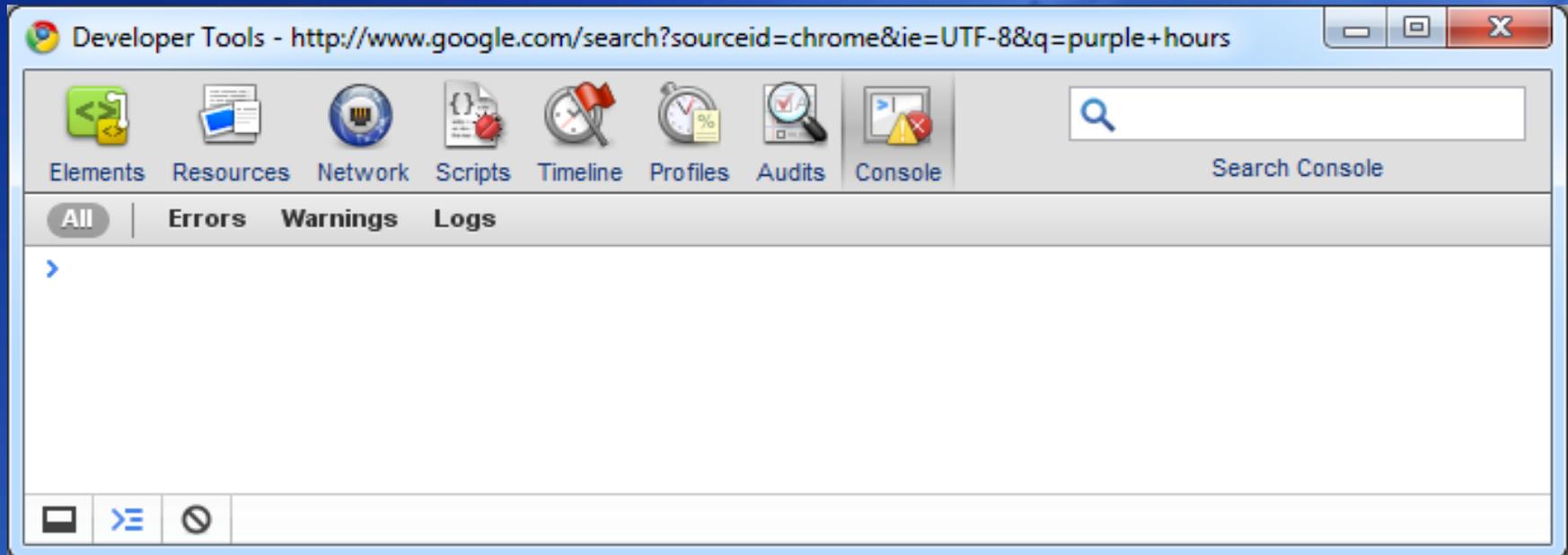
Inspect element

Show console

Dock to main window

Chrome Developer Tools (Cont.)

- Console Panel



Debugging Test for Chrome Developer Tools

The screenshot displays the Chrome Developer Tools interface for a file named "samplecode.html". The browser's address bar shows the file path: `file:///C:/Temp/samplecode.html`. A "Run" button is visible in the top left of the page content area.

The "Sources" panel is active, showing the following JavaScript code:

```
1 <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loos
2 <html>
3 <head>
4 <title> JavaScript Debugging Example </title>
5 <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
6 <script type="text/javascript">
7 //create a global variable for our <div>
8   var display;
9
10  function init() {
11    //initialize only after the HTML has been loaded
12    display = document.getElementById("results");
13  }
14
15
16  function firstParam() {
17    //set breakpoint here
18    var a = 5;
19    secondParam(a);
20
```

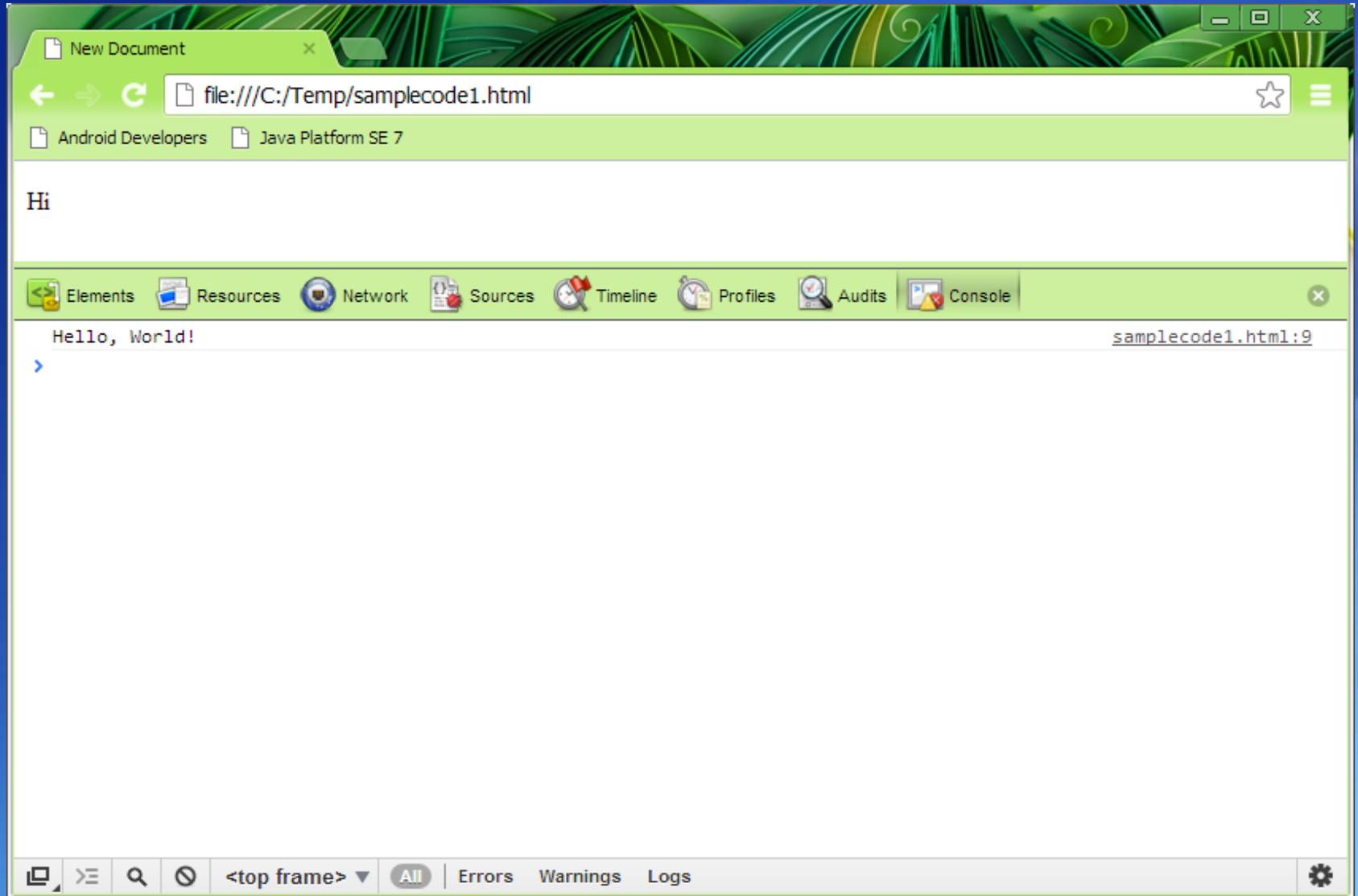
The right-hand side of the interface shows the "Call Stack" and "Scope Variables" panels, both indicating "Not Paused". The "Breakpoints" panel shows "No Breakpoints". The "DOM Breakpoints", "XHR Breakpoints", "Event Listener Breakpoints", and "Workers" panels are also visible.

The bottom status bar shows the current frame as "<top frame>" and the filter set to "All".

Debugging Test for Chrome Developer Tools (Cont.)

1. In Chrome Browser, load the example.
2. Press **Ctrl + Shift + I** to open the **Chrome Developer** tools, and click the **Sources** tab.
3. In the bottom pane, scroll to the first function, click the line that says "**var a = 5;**", and click **Breakpoints** tab in the right pane.
4. Click **F8** to process **Pause script execution**, and then click **Refresh** in the browser toolbar, and then click **F11** to process **Step into next function call**.
5. Click **Run** button in the web page, click the **Scope Variables** tab on the right panel.

Using console.log in Chrome



Firefox and Firebug

<http://getfirebug.com>

Firebug - Mozilla Firefox

File Edit View History Bookmarks Tools Help

Firebug

getfirebug.com

Google

What is Firebug?
Introduction and Features

Documentation
FAQ and Wiki

Community
Discussion forums and lists

Get Involved
Hack the code, create extensions

Firebug
Web Development Evolved.

Install Firebug
for Firefox, 100% free and open source

[Other Versions](#) [Firebug Lite](#) [Extensions](#)

The most popular and powerful web development tool

- ✓ Inspect HTML and modify style and layout in real-time
- ✓ Use the most advanced JavaScript debugger available for any browser
- ✓ Accurately analyze network usage and performance
- ✓ Extend Firebug and add features to make Firebug even more powerful
- ✓ Get the information you need to get it done with Firebug.

[More Features »](#)

Introduction to Firebug
Firebug pyroentomologist Rob Campbell gives a quick introduction to Firebug.
[Watch now »](#)

[More Screencasts »](#)

Inspect
Pinpoint an element in a webpage with ease and precision.

Log
Send messages to the console direct from your webpage through Javascript.

Profile
Measure your Javascript performance in the Console's Profiler.

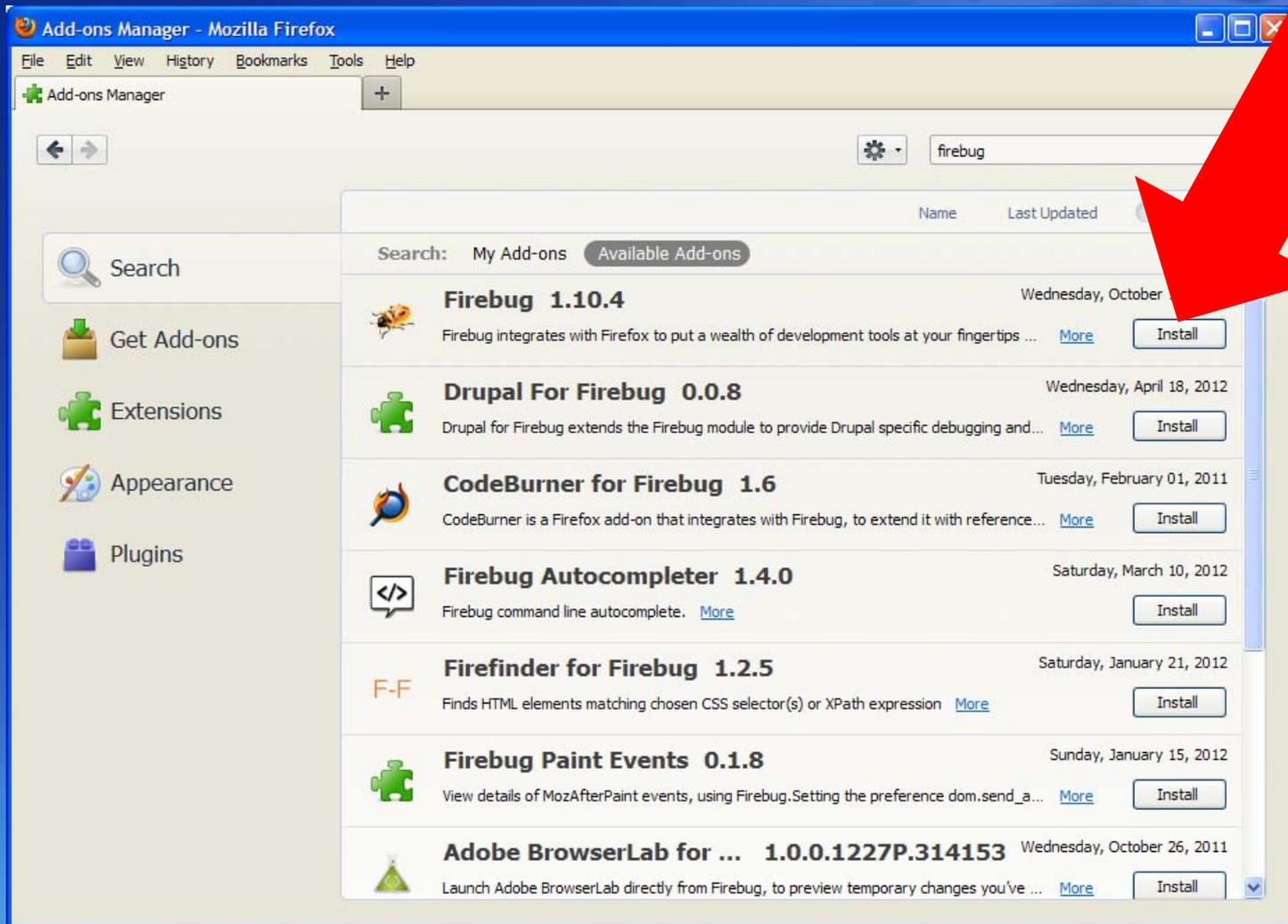
Debug
Step-by-step interactive debugging in a visual environment.

Analyze
Look at detailed measurements of your site's network activity.

Layout
Tweak and position HTML elements with CSS and the Layout panel.

Firefox and Firebug (Cont.)

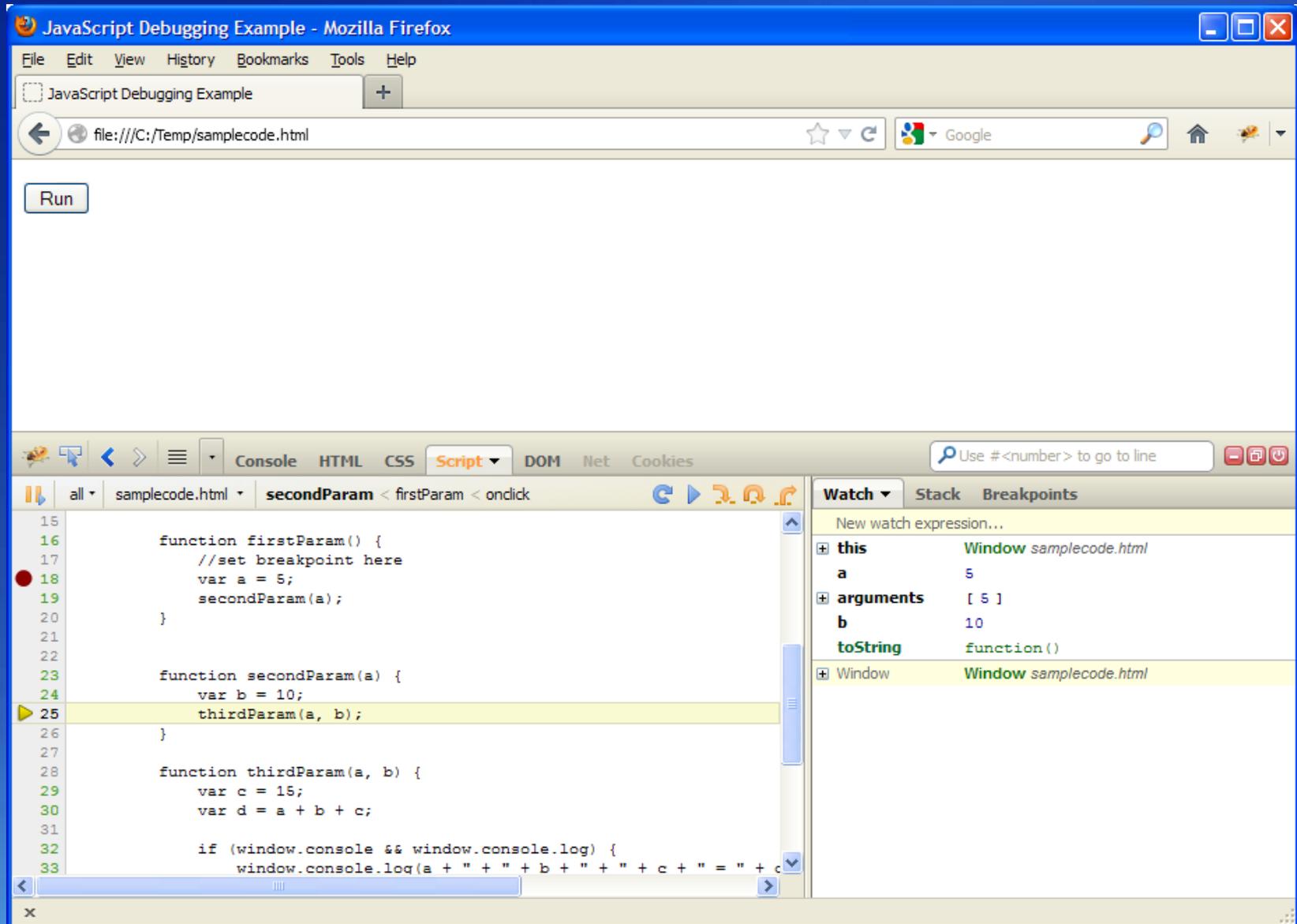
- Firebug Installation on Firefox.



The screenshot shows the Firefox Add-ons Manager window. The search bar at the top right contains the text 'firebug'. The search results are displayed in a table with columns for 'Name' and 'Last Updated'. The first result is 'Firebug 1.10.4', which is highlighted. A large red arrow points to the 'Install' button for this add-on. The other results include 'Drupal For Firebug 0.0.8', 'CodeBurner for Firebug 1.6', 'Firebug Autocompleter 1.4.0', 'Firefinder for Firebug 1.2.5', 'Firebug Paint Events 0.1.8', and 'Adobe BrowserLab for ... 1.0.0.1227P.314153'.

Name	Last Updated
Firebug 1.10.4 Firebug integrates with Firefox to put a wealth of development tools at your fingertips ... More	Wednesday, October 26, 2011
Drupal For Firebug 0.0.8 Drupal for Firebug extends the Firebug module to provide Drupal specific debugging and ... More	Wednesday, April 18, 2012
CodeBurner for Firebug 1.6 CodeBurner is a Firefox add-on that integrates with Firebug, to extend it with reference... More	Tuesday, February 01, 2011
Firebug Autocompleter 1.4.0 Firebug command line autocomplete. More	Saturday, March 10, 2012
Firefinder for Firebug 1.2.5 Finds HTML elements matching chosen CSS selector(s) or XPath expression More	Saturday, January 21, 2012
Firebug Paint Events 0.1.8 View details of MozAfterPaint events, using Firebug.Setting the preference dom.send_a... More	Sunday, January 15, 2012
Adobe BrowserLab for ... 1.0.0.1227P.314153 Launch Adobe BrowserLab directly from Firebug, to preview temporary changes you've ... More	Wednesday, October 26, 2011

Debugging Test for Firefox Firebug



The screenshot displays the Mozilla Firefox JavaScript Debugging interface. The browser window shows the file `file:///C:/Temp/samplecode.html`. The Firebug console is open, showing the `Script` tab with the following code:

```
15     function firstParam() {
16         //set breakpoint here
17         var a = 5;
18         secondParam(a);
19     }
20
21
22
23     function secondParam(a) {
24         var b = 10;
25         thirdParam(a, b);
26     }
27
28     function thirdParam(a, b) {
29         var c = 15;
30         var d = a + b + c;
31
32         if (window.console && window.console.log) {
33             window.console.log(a + " + " + b + " + " + c + " = " + c
```

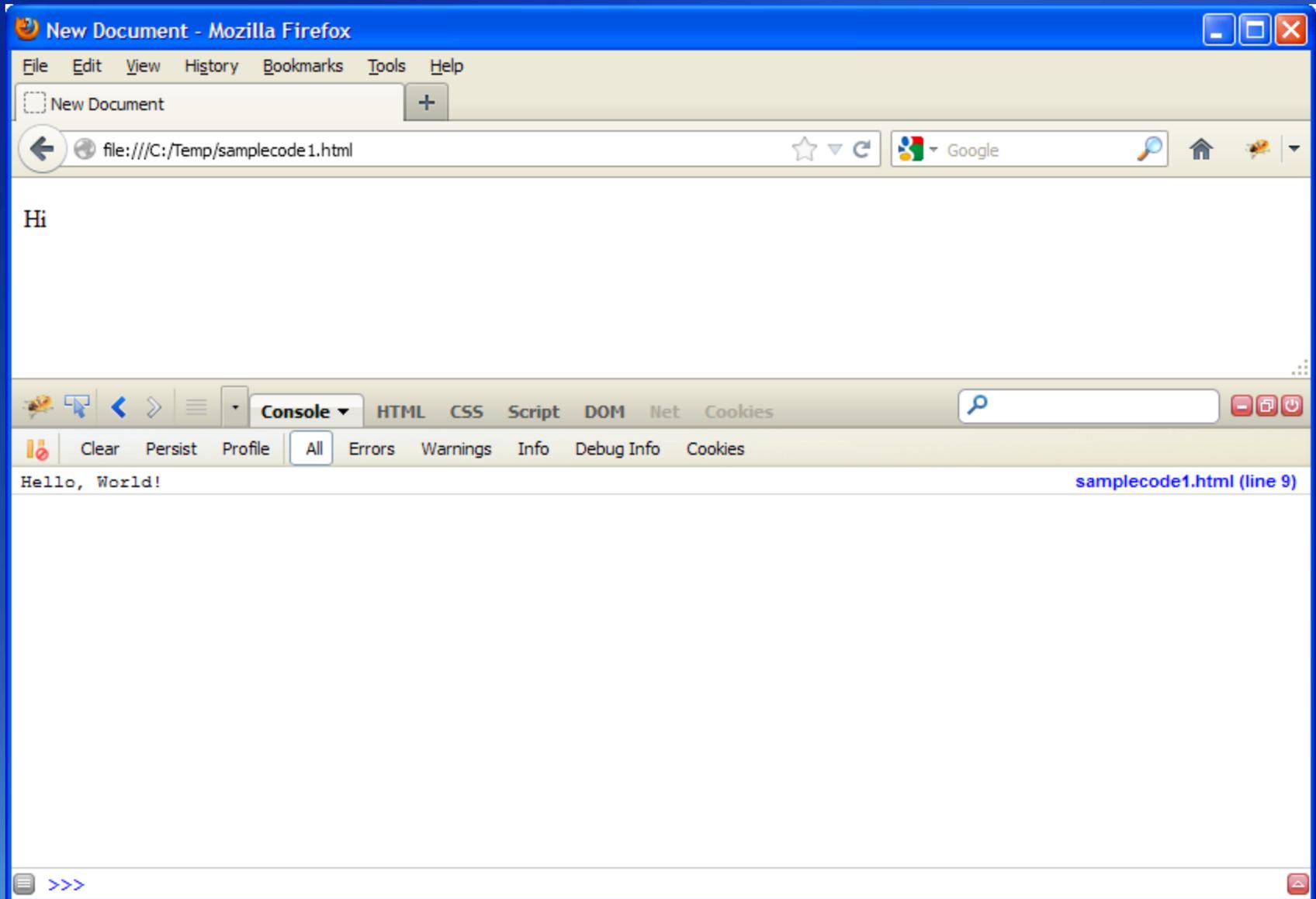
A red dot indicates a breakpoint is set at line 25. The `Watch` panel on the right shows the following variables and their values:

Variable	Value
this	Window samplecode.html
a	5
arguments	[5]
b	10
toString	function()
Window	Window samplecode.html

Debugging Test for Firefox Firebug (Cont.)

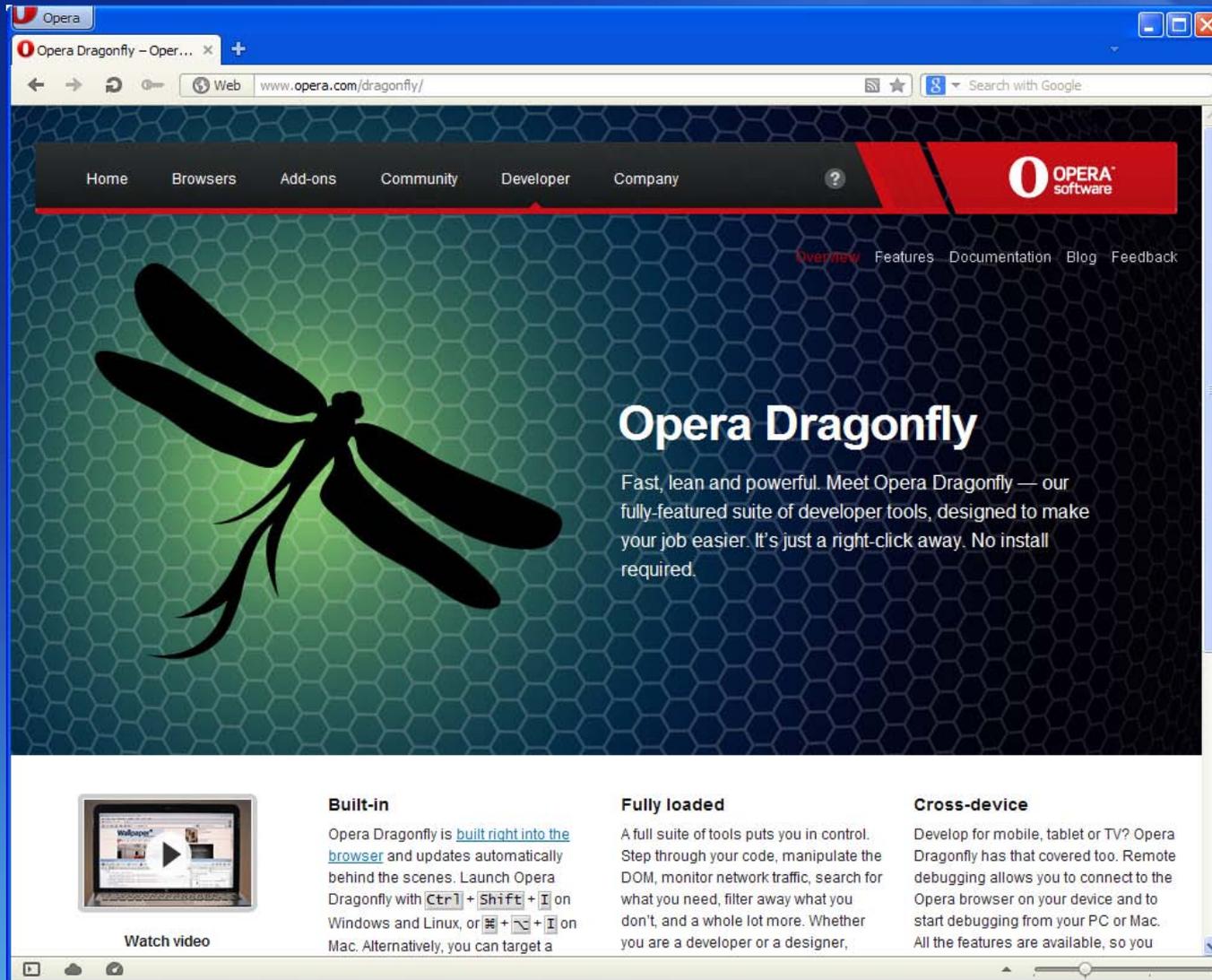
1. In Firefox Browser, load the example.
2. Press **F12** to open the **Open Firebug** tools, and click the **Script** tab.
3. In the bottom pane, scroll to the first function, click the line that says "**var a = 5;**".
4. Click **Run** button in the web page, click **F11** to process **Step Into**.
5. Click the **Watch** tab on the right panel.

Using console.log in Firefox



Opera and Dragonfly

• <http://www.opera.com/dragonfly/>



The screenshot shows the Opera Dragonfly website in a browser window. The browser's address bar displays "www.opera.com/dragonfly/". The website features a dark blue background with a hexagonal pattern. A large black dragonfly silhouette is positioned on the left side. The main heading "Opera Dragonfly" is centered, followed by the text: "Fast, lean and powerful. Meet Opera Dragonfly — our fully-featured suite of developer tools, designed to make your job easier. It's just a right-click away. No install required." Below this, there are three columns of text, each with a sub-heading and a paragraph. The first column includes a video player thumbnail and the text "Watch video". The second column is titled "Built-in" and describes how the tool is integrated into the browser. The third column is titled "Fully loaded" and lists various developer tools. The fourth column is titled "Cross-device" and discusses remote debugging capabilities. The Opera logo and "OPERA software" are visible in the top right corner of the website.

Opera

Opera Dragonfly – Oper... x

Web www.opera.com/dragonfly/ Search with Google

Home Browsers Add-ons Community Developer Company

OPERA software

Overview Features Documentation Blog Feedback

Opera Dragonfly

Fast, lean and powerful. Meet Opera Dragonfly — our fully-featured suite of developer tools, designed to make your job easier. It's just a right-click away. No install required.

Built-in

Opera Dragonfly is [built right into the browser](#) and updates automatically behind the scenes. Launch Opera Dragonfly with **Ctrl + Shift + I** on Windows and Linux, or **⌘ + ⌥ + I** on Mac. Alternatively, you can target a

Fully loaded

A full suite of tools puts you in control. Step through your code, manipulate the DOM, monitor network traffic, search for what you need, filter away what you don't, and a whole lot more. Whether you are a developer or a designer,

Cross-device

Develop for mobile, tablet or TV? Opera Dragonfly has that covered too. Remote debugging allows you to connect to the Opera browser on your device and to start debugging from your PC or Mac. All the features are available, so you

Watch video

Debugging Test for Opera Dragonfly

The screenshot shows the Opera Dragonfly JavaScript debugger interface. The browser window displays the URL `localhost/C:/Temp/samplecode.html`. The code editor shows the following JavaScript code:

```
1
2   var display;
3   function init() {
4       display = document.getElementById("results");
5   }
6   function firstParam() {
7       //set breakpoint here
8       var a = 5;
9       secondParam(a);
10  }
11  function secondParam(a) {
12      var b = 10;
13      thirdParam(a, b);
14  }
15  function thirdParam(a, b) {
16      var c = 15;
17      var d = a + b + c;
18      if (window.console && window.console.log) {
19          window.console.log(a + " + " + b + " + " + c + " = " + d);
20      } else {
21          display.innerHTML = a + " + " + b + " + " + c + " = " + d;
22      }
23  }
24
```

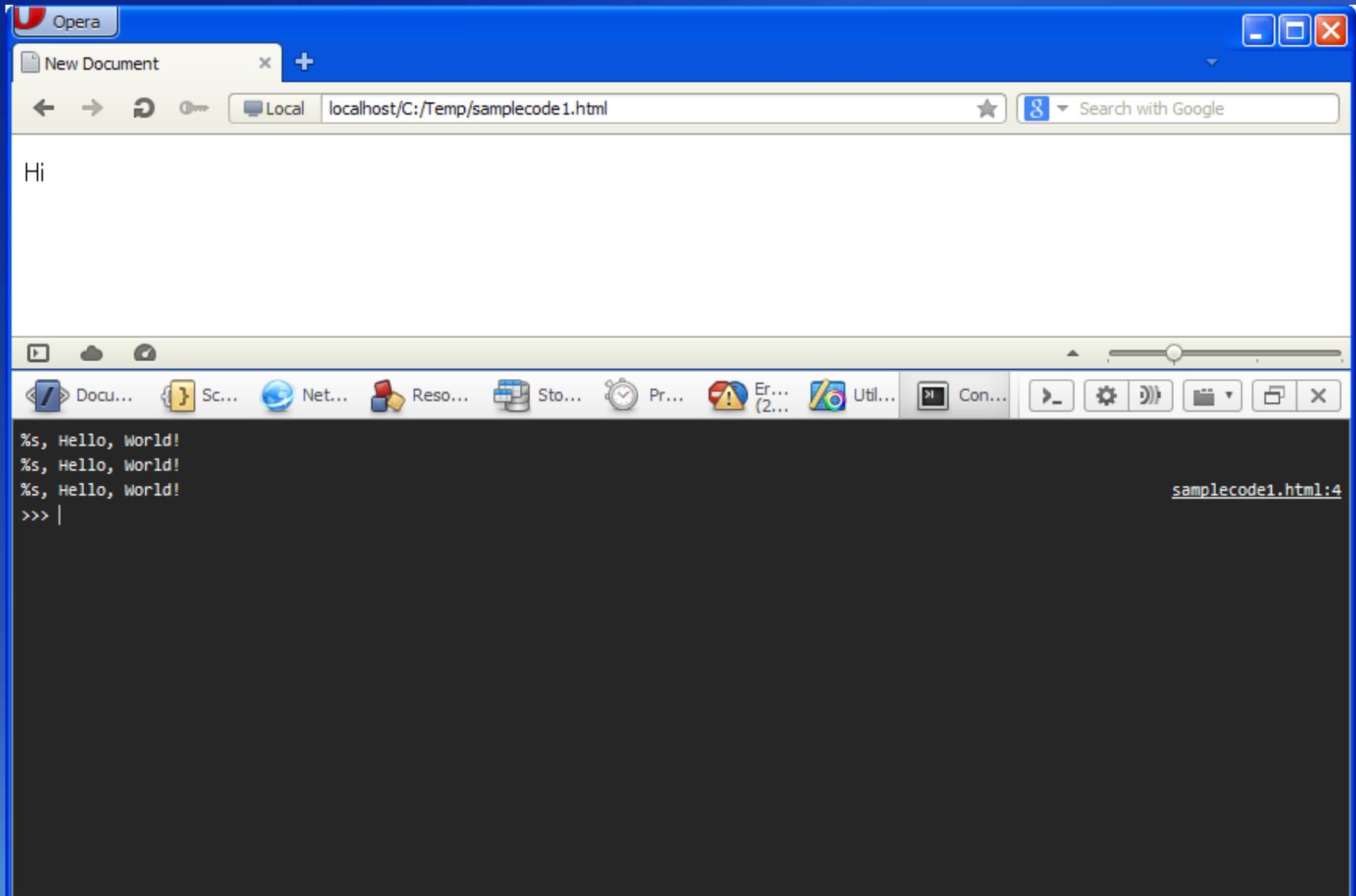
The debugger is currently paused at line 16. The right-hand side of the interface shows the following panels:

- State**: Breakpoints, Search
- Watches**: No return values
- Return Values**: No return values
- Call Stack**:
 - thirdParam samplecode.html:16
 - secondParam samplecode.html:13
 - firstParam samplecode.html:9
 - (anonymous) samplecode.html:1
 - (global)
- Inspection**:
 - this window
 - a 5
 - arguments Arguments
 - b 10
 - c undefined
 - d undefined
- Scope Chain**

Debugging Test for Opera Dragonfly (Cont.)

1. In Opera Browser, load the example.
2. Press **Ctrl + Shift + I** to open the **Opera Dragonfly** tools, and click the **Scripts** tab.
3. In the bottom pane, scroll to the first function, click the line that says "**var a = 5;**".
4. Click **Run** button in the web page, click **F11** to process **Step Into**.
5. Click the **Inspection** tab on the right panel.

Using console.log in Opera



IE's F12 Developer Tools

The screenshot displays the Internet Explorer Developer Tools interface. The main window shows the source code of a JavaScript debugging example. A breakpoint is set on line 13, where the variable `a` is assigned the value 5. The Locals pane on the right shows the current state of variables: `a` is 5, `b` is 10, `c` is 15, and `d` is undefined.

```
1 <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/strict.dtd">
2 <html>
3 <head>
4 <title> JavaScript Debugging Example </title>
5 <meta http-equiv="Content-Type" content="text/html" />
6 <script type="text/javascript">
7     var display;
8     function init() {
9         display = document.getElementById("results");
10    }
11    function firstParam() {
12        //set breakpoint here
13        var a = 5;
14        secondParam(a);
15    }
16    function secondParam(a) {
17        var b = 10;
18        thirdParam(a, b);
19    }
20    function thirdParam(a, b) {
21        var c = 15;
22        var d = a + b + c;
23        if (window.console && window.console.log)
24            window.console.log(a + " + " + b + " = " + c);
25        } else {
26            display.innerHTML = a + " + " + b + " = " + c;
27        }
28    }
29 </script>
30 </head>
31 <body onload="init()">
32     <p><button onclick="firstParam();">Run</button>
33     <div id="results"></div>
34 </body>
35 </html>
```

Name	Value	Type
a	5	Number
b	10	Number
c	15	Number
d	undefined	Undefined

Debugging Test for IE (Cont.)

1. In Internet Explorer 9, load the example.
2. Press **F12** to open the **F12** tools, and click the **Script** tab.
3. In the left pane, scroll to the first function, right-click the line that says "var a = 5;", and click **Insert breakpoint**.
4. Click **Start debugging**, and then on the webpage in the browser, click the **Run** button.
5. In **F12** tools, click the **Watch** tab on the right side, and add the variables "a, b, c, and d."
6. Step through your code by pressing **F11**, or by clicking the **Step into** button, and watch the variables on the **Watch** tab.

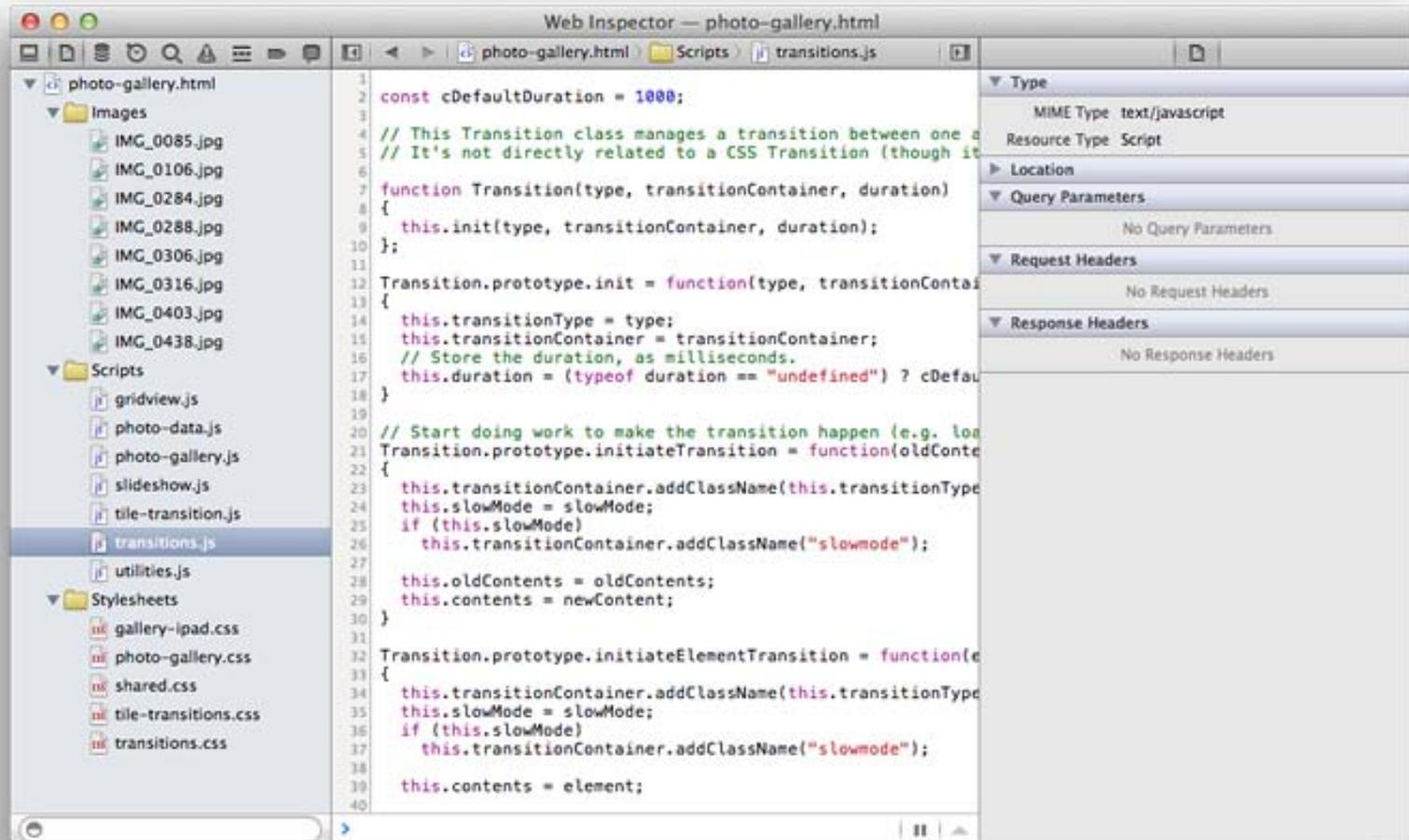
Using console.log in IE

The screenshot shows the Internet Explorer Developer Tools window open over a new document. The document contains the following HTML and JavaScript code:

```
1 <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Trans
2 <html>
3 <head>
4 <title> New Document </title>
5 <meta http-equiv="Content-Type" content="text/h
6 <script type="text/javascript">
7   function hello() {
8     var msg = "Hello, World!";
9     console.log("%s", msg);
10  }
11 </script>
12 </head>
13 <body onload="hello()">
14   <p>Hi</p>
15 </body>
16 </html>
```

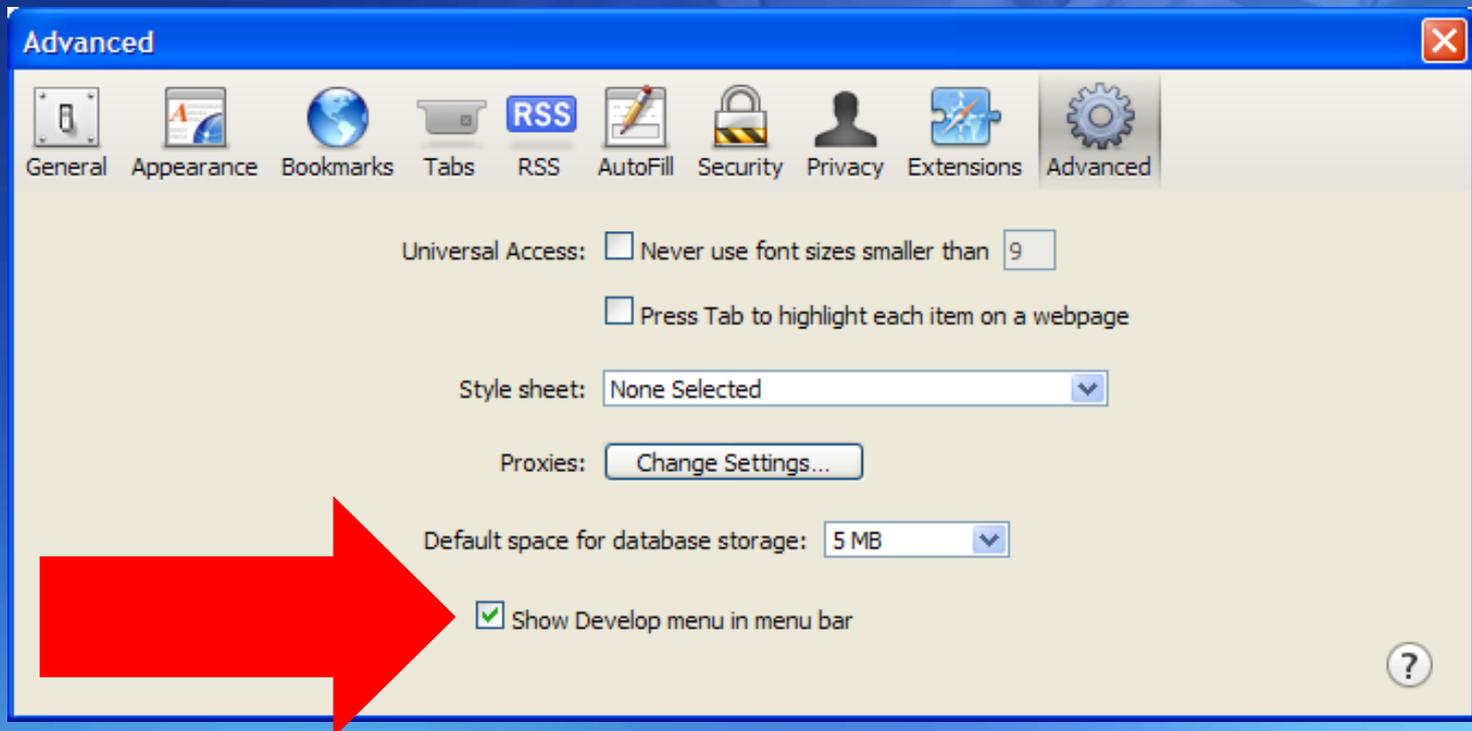
The Developer Tools window is set to the 'Script' tab, and the 'Console' pane is visible on the right. The status bar at the bottom of the Developer Tools shows 'Run Script' and 'Multi Line Mode' buttons.

Safari Web Development Tools

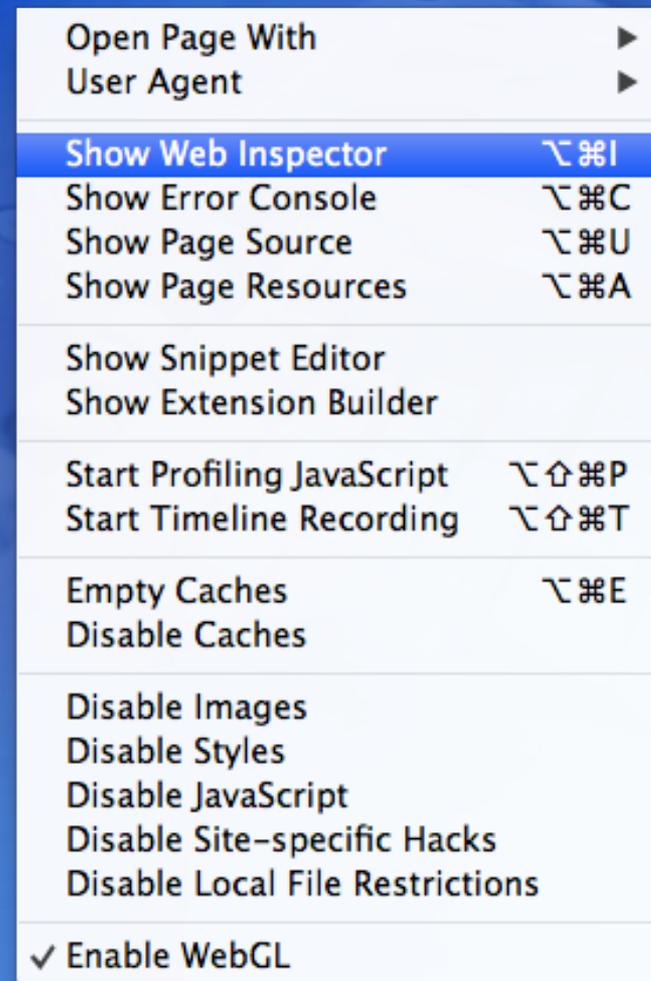
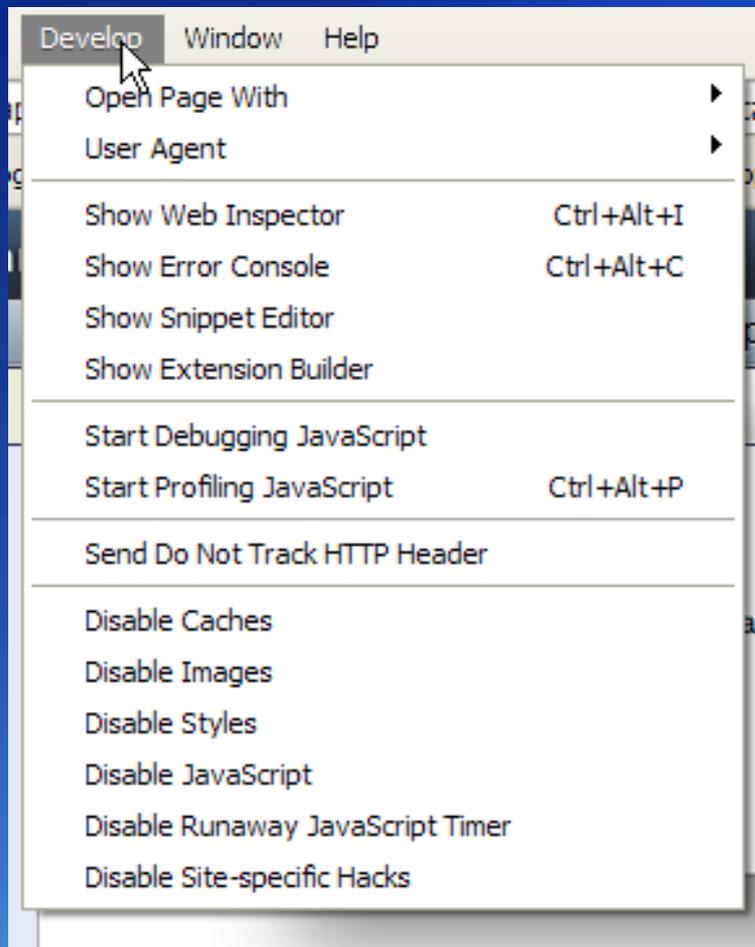


Safari Web Development Tools (Cont.)

1. In Safari **preferences**, click **Advanced**, then select "**Show Develop menu in menu bar**"



Safari Web Development Tools (Cont.)



Debugging Test for Safari

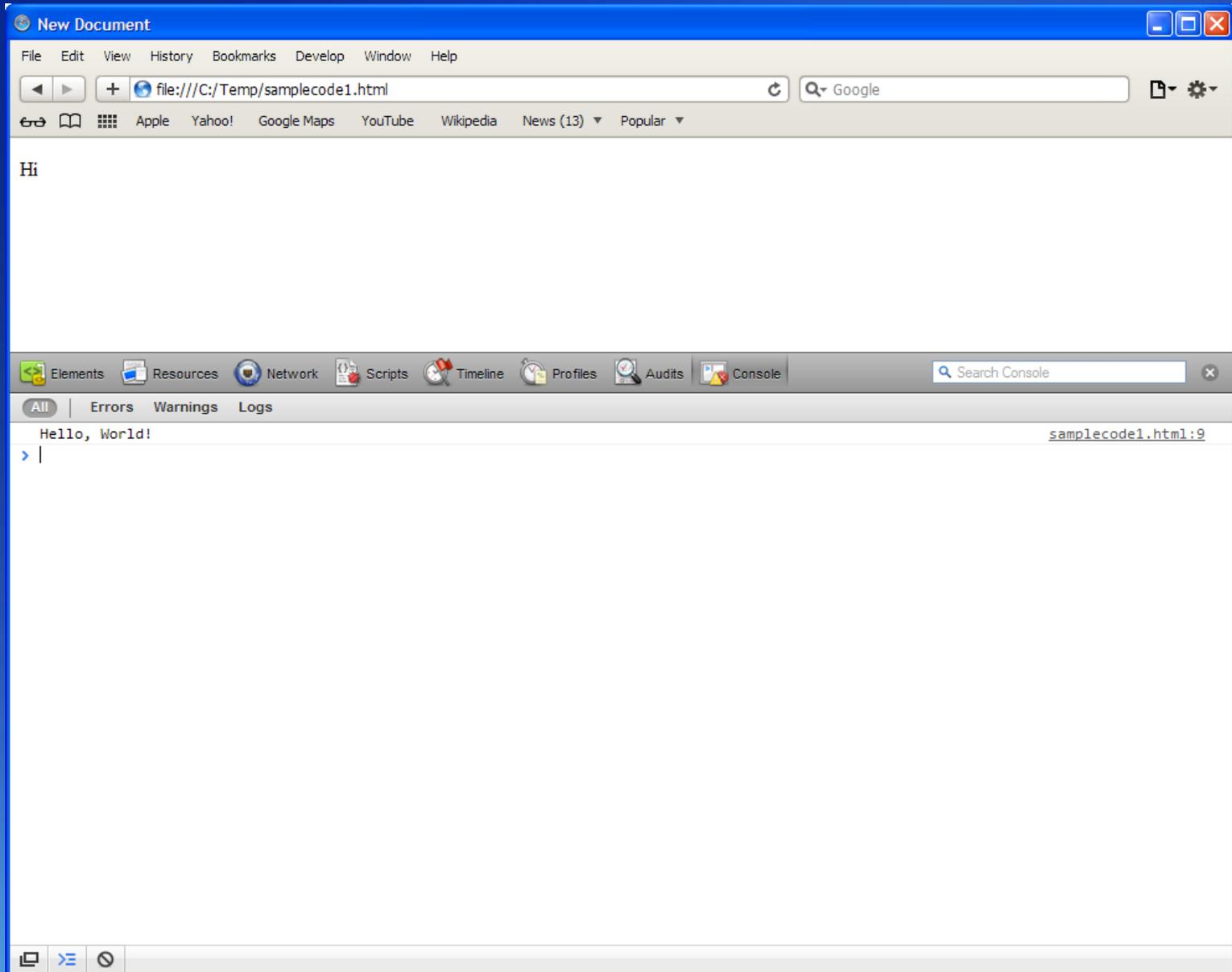
The screenshot displays the Safari JavaScript Debugging interface. The main window shows the source code of a file named 'samplecode.html'. A blue arrow indicates a breakpoint is set at line 13, which contains the code `var a = 5;`. The right-hand sidebar contains several debugging panels: 'Watch Expressions', 'Call Stack', 'Scope Variables', 'Breakpoints', and 'Workers'. The 'Breakpoints' panel is expanded, showing the active breakpoint at 'samplecode.html:13' with the variable 'var a = 5;'. The 'Workers' panel has a 'Debug' checkbox. The code editor shows the following JavaScript code:

```
1 <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
2 <html>
3 <head>
4 <title> JavaScript Debugging Example </title>
5 <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
6 <script type="text/javascript">
7   var display;
8   function init() {
9     display = document.getElementById("results");
10  }
11  function firstParam() {
12    //set breakpoint here
13    var a = 5;
14    secondParam(a);
15  }
16  function secondParam(a) {
17    var b = 10;
18    thirdParam(a, b);
19  }
20  function thirdParam(a, b) {
21    var c = 15;
22    var d = a + b + c;
23    if (window.console && window.console.log) {
24      window.console.log(a + " + " + b + " + " + c + " = " + d);
25    } else {
26      display.innerHTML = a + " + " + b + " + " + c + " = " + d;
27    }
28  }
29 </script>
30 </head>
31 <body onload="init()">
```

Debugging Test for Safari (Cont.)

1. In Safari Browser, load the example.
2. Press **Show Web Inspector** in **Develop** Menu to open the **Safari Development Tool** tools, and click the **Scripts** tab.
3. In the bottom pane, scroll to the first function, click the line that says "**var a = 5;**", and click **Breakpoints** tab in the right pane.
4. Click **Pause script execution** button, and then click **Refresh** in the browser toolbar, and then click **Step into next function call** button on the right panel.
5. Click **Run** button in the web page, click the **Scope Variables** tab on the right panel.

Using console.log in Safari



Dealing with Cross-Browser Differences

- 15 years ago, dealing with cross-browser differences between the two dominant browsers at the time—Internet Explorer and Netscape’s Navigator—was a heroic and painful process.
- Differences still exist, and all of the differences aren’t just between Firefox-Opera-Safari-Chrome and Internet Explorer.

Object Detection

```
<script type="text/javascript">
  // *** BROWSER VERSION ***
  this.major = parseInt(navigator.appVersion)
  this.minor = parseFloat(navigator.appVersion)
  this.nav = ((agt.indexOf('mozilla')!=-1) && ((agt.indexOf('spoofer')== -1)
  && (agt.indexOf('compatible') == -1)))
  this.nav2 = (this.nav && (this.major == 2))
  this.nav3 = (this.nav && (this.major == 3))
  this.nav4 = (this.nav && (this.major == 4))
  this.nav4up = this.nav && (this.major >= 4)
  this.navonly = (this.nav && (agt.indexOf(";nav") != -1))
  this.ie = (agt.indexOf("msie") != -1)
  this.ie3 = (this.ie && (this.major == 2))
  this.ie4 = (this.ie && (this.major == 4))
  this.ie4up = this.ie && (this.major >= 4)
  this.opera = (agt.indexOf("opera") != -1)
</script>
```

Object Detection (Cont.)

- With object detection, the JavaScript application accesses the object being detected in a conditional statement.
- If the object doesn't exist, the condition evaluates to **false**.
- As an example, the following checks to ensure that the most basic level of object model support is provided in the browser or other user agent.

```
if (document.getElementById) . . .
```

Object Detection (Cont.)

```
<script type="text/javascript">
  function showBlock(evt) {
    var theEvent = evt ? evt : window.event;
    var theSrc = theEvent.target ? theEvent.target : theEvent.srcElement;
    var itemId = "elements" + theSrc.id.substr(5,1);
    var item = document.getElementById(itemId);
    if (item.style.display==='none') {
      item.style.display='block';
    } else {
      item.style.display='none';
    }
  }
</script>
```

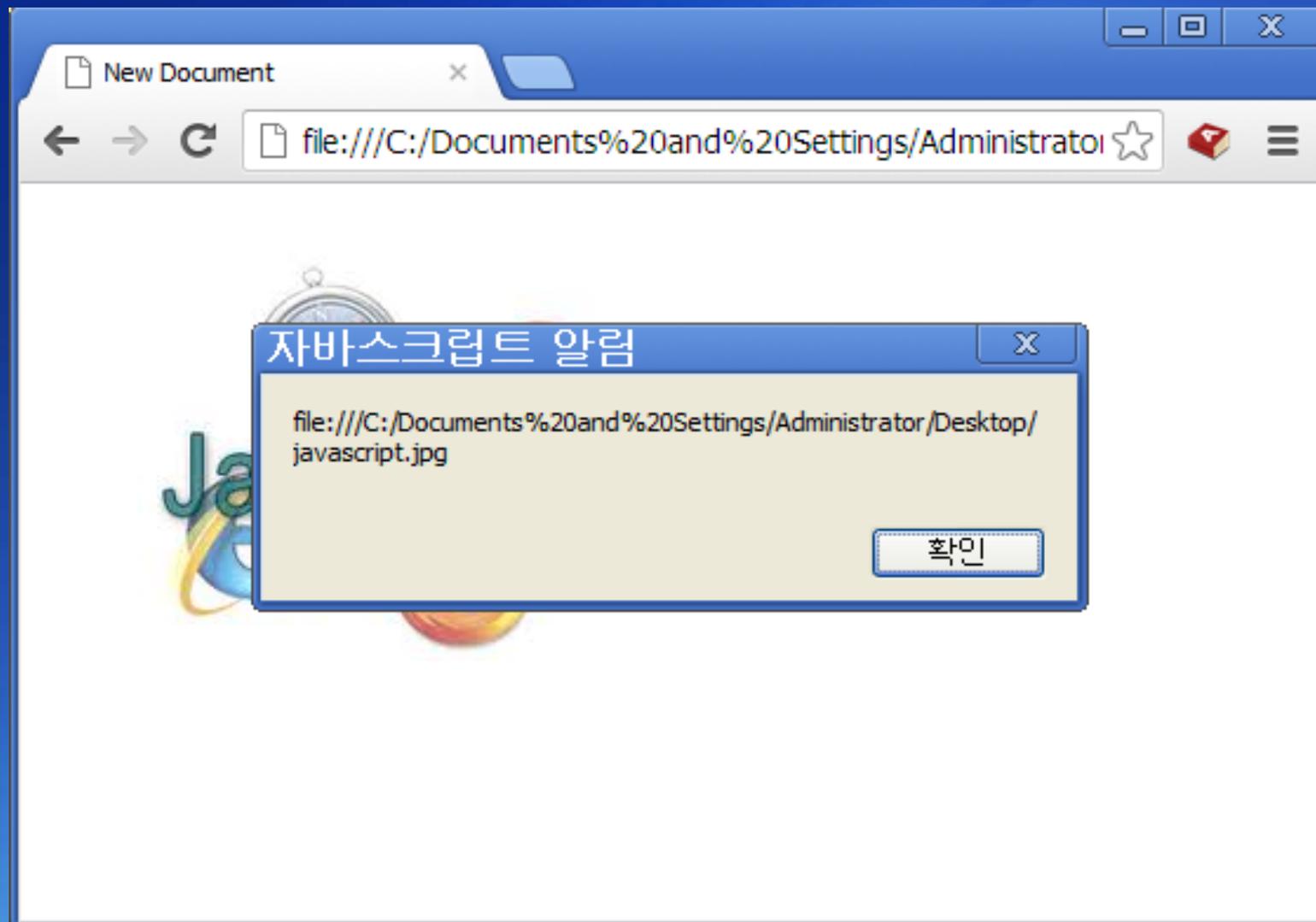
Object Detection (Cont.)

```
<style type="text/css">
  div {
    position: absolute;
    top: 30px;
    left: 50px;
  }
  #div1 img {
    filter:
    progid:DXImageTransform.Microsoft.AlphaImageLoader(src=javascript.jpg,
    sizingMethod='scale');
  }
</style>
```

```
<script type="text/javascript">
  window.onload=function() {
    document.getElementById("div1").onclick=getSrc;
  }
  function getSrc(evt) {
    var theEvent = evt ? evt : window.event;
    var theSrc = theEvent.target ? theEvent.target : theEvent.srcElement;
    alert(theSrc.src);
  }
</script>
```

```
<body>
  <div id="div1">
    
  </div>
</body>
```

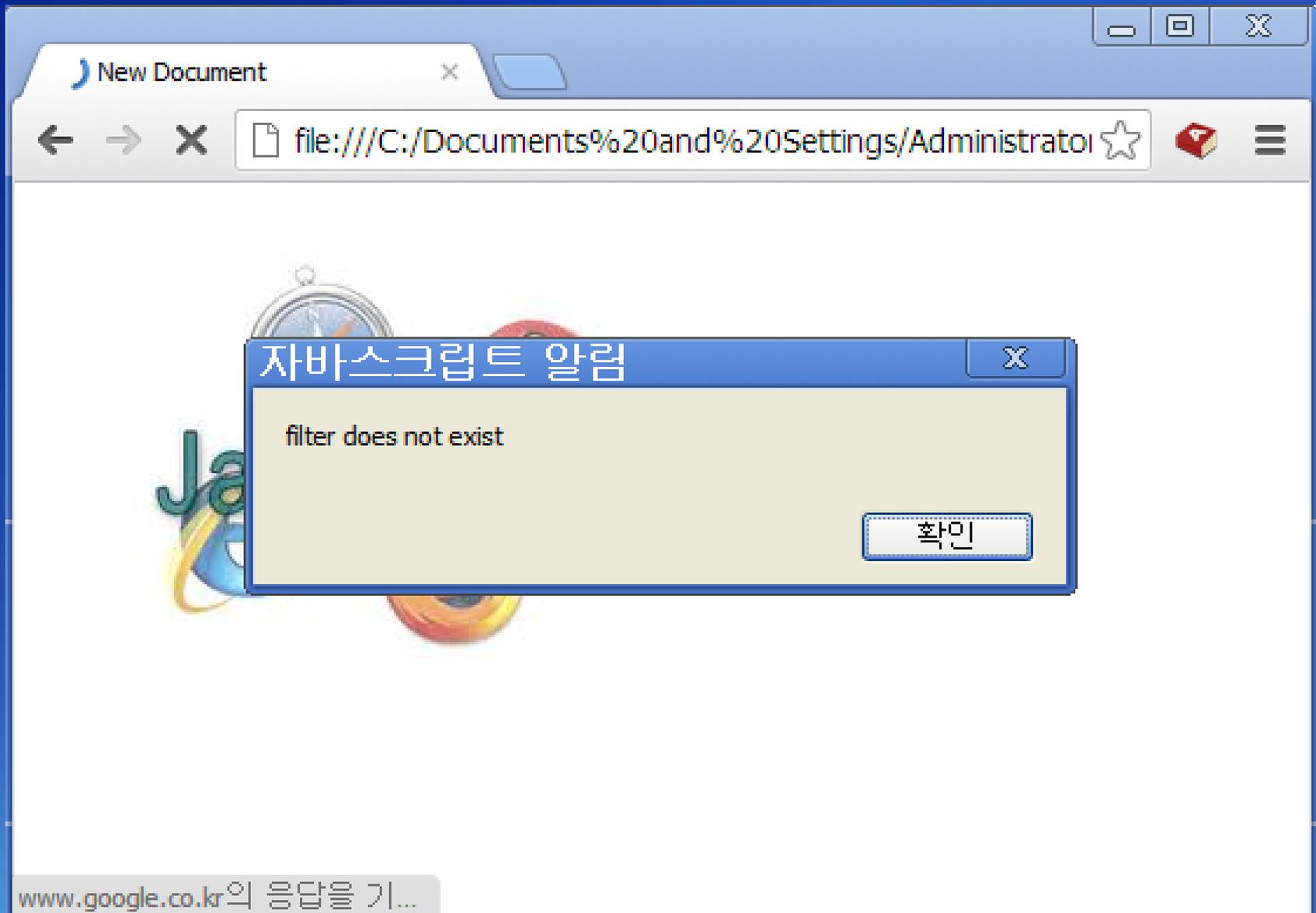
Object Detection (Cont.)



Object Detection (Cont.)

```
<script type="text/javascript">
window.onload=function() {
  var divElement = document.getElementById("div1");
  var tst1 = (divElement.style.filter) ? "filter exists" : "filter does not exist";
  alert(tst1);
  var tst2 = (typeof(divElement.style.filter) !== "undefined") ? "filter exists" : "filter does
not exist";
  alert(tst2);
  var tst3 = (divElement.style.filter !== undefined) ? "filter exists" : "filter does not exist";
  alert(tst3);
  var tst4 = (divElement.style.opacity) ? "opacity exists" : "opacity does not exist";
  alert(tst4);
  var tst5 = (typeof(divElement.style.opacity) !== "undefined") ? "opacity exists" :
"opacity does not exist";
  alert(tst5);
}
</script>
```

Object Detection (Cont.)



Test Your Knowledge : Quiz

- 1.** Write the code to check to see what the value of a variable is, without recourse to browser-specific debugger.

Test Your Knowledge : Quiz

1. Write the code to check to see what the value of a variable is, without recourse to browser-specific debugger.

```
// test some variable  
alert(firstName);  
// value will either be set, null, or undefined
```

Test Your Knowledge : Quiz (Cont.)

- 2.** The CSS attribute `text-shadow` has an interesting history. It was added to CSS2, but no browser implemented it, and it was removed in CSS 2.1. However, then browsers implemented it, and now it's being added into CSS3. At the time of this writing, this attribute was implemented in two of our four target browsers. Can you use object detection to successfully test for this style attribute? Write cross browser code that sets this CSS attribute for an existing header element.

Test Your Knowledge : Quiz (Cont.)

2. The CSS attribute `text-shadow` has an interesting history. It was added to CSS2, but no browser implemented it, and it was removed in CSS 2.1. However, then browsers implemented it, and now it's being added into CSS3. At the time of this writing, this attribute was implemented in two of our four target browsers. Can you use object detection to successfully test for this style attribute? Write cross browser code that sets this CSS attribute for an existing header element.

```
var headerElement = document.getElementById("pageHeader");  
headerElement.style.textShadow="#ff0000 2px 2px 3px";
```