Notice for Copying JavaScript Code from these Slides

• When copying any JavaScript code from these slides, the console might return the following:

```javascript
>>> ['Harman', 'Goei', 'CSCI', '571'];
SyntaxError: illegal character
```

• To fix this, **retype** each quote.
Outline

• **What is Firebug?** Why do web developers use Firebug?
• **Installation of Firebug for Mozilla Firefox browser**
• **Launching Firebug for the First Time**
• **The Panels of Firebug**
• **Firebug Tutorials**
What is Firebug?

- Firebug is an extension for the Mozilla Firefox browser that allows you to debug and inspect HTML, CSS, the Document Object Model (DOM) and JavaScript.
Why do Web Developers use Firebug?

• Inspect the behavior of HTML/CSS, and modify style & layout with true WYSIWYG
• Debug JavaScript
• Detect performance of website
• Track Cookies & Sessions
Installation of Firebug for Mozilla Firefox Browser

• To install Firebug for Firefox, go to http://www.getfirebug.com (click on Install Firebug)

• Don't have Firefox? Firebug has a lite version which can be saved as a bookmark or embedded into your web page in JavaScript.
2 Ways to Launch Firebug

With the Mozilla Firefox browser open...

1) Press **F12** on the keyboard
   *(By default, body HTMLElement is selected)*

**OR**

2) Press the Firebug button on the toolbar
   *(By default, body HTMLElement is selected)*

*Note: Firebug may continue from a last saved session if it is still running*
The 3rd Way to Launch Firebug

1) In the current webpage, **right click** on an element (an image, text, background, etc).

2. In the dropdown menu, click on

   **Inspect Element with Firebug**

   *(the element you right clicked on is selected.)*
Hello, Firebug!

Let's take a look at the Firebug Toolbar first, as we will use this throughout the tutorial.
**Firebug Toolbar**

**PANELS**

- **Console**: Brings up a Interactive JavaScript Console
- **HTML**: Brings up the HTML View (see previous)
- **CSS**: Brings up the CSS View
- **Script**: Brings up the JavaScript Debugger (used later)
- **DOM**: A list of all the DOM Properties *(defaults to window object)*
- **Net**: Displays requests made from the browser
- **Cookies**: Displays sessions & cookies from the browser
Firebug Toolbar – Firebug Button

1. The Firebug Button
   a. Hide Firebug (hides the panel)
   b. Deactivate Firebug (turns off Firebug)
   c. Firebug UI Location…
      a. Detached
      b. Left/Right
      c. Top/Bottom
   d. Open with Editor
   e. Options
   f. Firebug Online
   g. Customize Shortcuts
   h. About

Bold: Useful things in Firebug
To undock Firebug, click on 📦
Firebug UI Locations - Undocked

To dock Firebug, click on 🎨
List of Useful Firebug Shortcuts

<table>
<thead>
<tr>
<th>Action</th>
<th>Shortcut</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clear Console</td>
<td>F1</td>
</tr>
<tr>
<td>Deactivate Firebug</td>
<td>Shift+F12</td>
</tr>
<tr>
<td>Decrease Text Size</td>
<td>Ctrl+-</td>
</tr>
<tr>
<td>Open Firebug in New Window</td>
<td>Ctrl+F12</td>
</tr>
<tr>
<td>Focus Command Line</td>
<td>Ctrl+Shift+l</td>
</tr>
<tr>
<td>Focus Firebug Search</td>
<td>Ctrl+f</td>
</tr>
<tr>
<td>Focus Location</td>
<td>Ctrl+Shift-space</td>
</tr>
<tr>
<td>Focus Watch Editor</td>
<td>Ctrl+Shift+n</td>
</tr>
<tr>
<td>Open help</td>
<td>F1</td>
</tr>
<tr>
<td>Increase Text Size</td>
<td>Ctrl++</td>
</tr>
<tr>
<td>Switch to left Firebug panel</td>
<td>Ctrl+Shift-Page Up</td>
</tr>
<tr>
<td>Go Back</td>
<td>Ctrl+Shift-Left Arrow</td>
</tr>
<tr>
<td>Go Forward</td>
<td>Ctrl+Shift-Right Arrow</td>
</tr>
<tr>
<td>Next Object</td>
<td>Ctrl+0</td>
</tr>
<tr>
<td>Normal Text Size</td>
<td>Ctrl+0</td>
</tr>
<tr>
<td>Previous Firebug panel</td>
<td>Ctrl-1</td>
</tr>
<tr>
<td>Previous Object</td>
<td>Ctrl+,</td>
</tr>
<tr>
<td>Re-enter Command</td>
<td>Ctrl+Shift-c</td>
</tr>
<tr>
<td>Remove All Cookies</td>
<td>Ctrl+Shift-o</td>
</tr>
<tr>
<td>Switch to right Firebug panel</td>
<td>Ctrl+Shift-Page Down</td>
</tr>
<tr>
<td>Toggle Break On ...</td>
<td>Ctrl+Alt+b</td>
</tr>
<tr>
<td>Open Firebug</td>
<td>F12</td>
</tr>
<tr>
<td>Toggle Inspecting</td>
<td>Ctrl+Shift-c</td>
</tr>
<tr>
<td>Toggle Profiling</td>
<td>Ctrl+Shift-p</td>
</tr>
<tr>
<td>Toggle Quick Info Box</td>
<td>Ctrl+Shift-i</td>
</tr>
</tbody>
</table>
2. **Inspect Element** (this is similar to Right click & Inspect Element with Firebug)

**The difference:** When hovering over elements in the page, the element is highlighted.

Also, the element is also highlighted in blue in the HTML View.

```
<div class="bigfirebug"></div>
```
Firebug Toolbar – Arrows, Quick Console, Show Hide Panels

3. Back/Forward – Switches between Panels

4. Quick Console – Interactive JavaScript console

5. Show or Hide Panels – Show or hide all panels
Firebug Tutorials Outline

1. **Inspecting HTML/CSS elements and their properties**
2. **Modifying HTML/CSS elements in real time**
3. **Debugging JavaScript & Analyzing Behavior of JavaScript code**
4. **Web Performance**
5. **Cookies and Sessions**
Inspecting HTML/CSS Elements and their Properties

1) Click here for the tutorial:
http://www-scf.usc.edu/~goei/571-firebug/lesson1.html

TUTORIAL OBJECTIVES
We are given the following element:

1) Find out the attributes & DOM properties of the element
2) Find out the computed CSS properties
3) Draw the box model for the element.
4) When hovering the element, determine the CSS properties.

Hovered

INSPECTING HTML/CSS ELEMENTS
AND THEIR PROPERTIES
Finding the Attributes & DOM properties of the Element

1) Use the Inspect Element feature in Firebug. (**Right click** on the element, and click on **Inspect Element with Firebug**)

**Note:** We can also use this button and click on the element
Finding the Attributes & DOM properties of the Element

2) The HTML Panel is displayed with the element selected.

From the highlighted area, we know the following about the element:

```html
<a class="btn btn-primary btn-large">I'm a cool button.</a>
```

INSPECTING HTML/CSS ELEMENTS AND THEIR PROPERTIES
Finding the Attributes & DOM properties of the Element

From the highlighted area & the breadcrumb we know the following about the element:

\[\text{\text{<a class="btn btn-primary btn-large">I'm a cool button.</a>}}\]

1. The element is an anchor
2. The element has a class attribute `btn btn-primary btn-large`
3. The element has a TextNode with TextContent “I’m a cool button”
4. The element's parent node is body (whose parent node is html)

There’s a lot more we can say about the element though!
Finding the Attributes & DOM properties of the Element

4) Click on the DOM panel on the CSS sidebar on the HTML view.
Finding the Attributes & DOM properties of the Element

A list of all DOM properties are displayed for the current element selected.

Notice: We have an attributes member variable, which has an array of attributes. This is another way how to determine the attributes for the selected element.
Find the Computed CSS Properties for that Element

1) Click on Style in the right hand column.
A list of CSS properties are shown, from most relevant to least.

What we know from the diagram:
1) List of CSS properties
   a) crossed out – It means the style was cascaded
2) Where it is located in the server
   - Which file,
   - What line number does the CSS property start
   - Clicking on lesson1.html (line 50) brings up the file in the CSS panel.

Tip: If you did not mean for a css property to be cascaded, you can simply add an !important to the cascaded property, or you must rearrange the CSS properties (files) (bottom CSS cascades the top CSS)
Find the Computed CSS Properties for that Element

2) Click on Computed
A list of Computed CSS properties are displayed.

Tip: Clicking on the +/- displays the computation for the CSS property (Cascaded properties are crossed out)

Next objective, finding the box model of the element

INSPECTING HTML/CSS ELEMENTS AND THEIR PROPERTIES
Find the Computed CSS Properties for that Element

1) Stay on the Computed panel and scroll to box model.
A list of Box Model CSS properties are shown.

We need to draw out the box model, however...

INSPECTING HTML/CSS ELEMENTS AND THEIR PROPERTIES
Find the Computed CSS Properties for that Element

2) Click on Layout
The Box Model is drawn out. When highlighting over an area, the browser highlights the selected element’s box property & displays a ruler
Purple highlight: Padding
Yellow highlight: Margin

INSPECTING HTML/CSS ELEMENTS AND THEIR PROPERTIES

USC Viterbi
School of Engineering
Determining Hover CSS Properties of the Element

1) Hover over the element.
The Style Panel will change accordingly.

Added CSS properties caused by the psuedo-class hover:

```css
.btn-primary:hover, .btn-primary:active,
.btn-primary.active, .btn-primary.disabled,
.btn-primary[disabled] {
  background-color: #0044CC;
  color: #FFFFFF;
}
```

INSPECTING HTML/CSS ELEMENTS AND THEIR PROPERTIES
Tutorial 1 Summary

TUTORIAL OBJECTIVES
1) Find out the attributes & DOM properties of the element
   Solution: In the HTML Panel, click DOM on the right side pane.

2) Find out the computed CSS properties
   Solution: In the HTML Panel, click DOM on the right side pane

3) Draw the box model for the element.
   Solution: In the HTML Panel, click on Layout

INSPECTING HTML/CSS ELEMENTS AND THEIR PROPERTIES
Tutorial 1 Summary

TUTORIAL OBJECTIVES

4) Determining CSS Hover Properties of the element
Solution: Hover over the element, watch the Style pane in HTML View

HOW IS THIS USEFUL?

1) Can visualize how an element is formed in CSS/HTML/JavaScript

Example – A client of yours likes a button at Google+. The client wants that same exact button in their web application. **Time to use Firebug.**
Modifying the DOM of an Element in Real Time

Go to http://www-scf.usc.edu/~goei/571-firebug/lesson2.html

TUTORIAL OBJECTIVES
Transform this page:

I am a paragraph element with black text and a white background.

The University of Southern California does not screen or control the content on this website and thus does not guarantee the accuracy, integrity, or quality of such content. All content on this website is provided by and is the sole responsibility of the person from which such content originated, and such content does not necessarily reflect the opinions of the University administration or the Board of Trustees.

TO

I am a paragraph element with white and a black background.

The University of Southern California does not screen or control the content on this website and thus does not guarantee the accuracy, integrity, or quality of such content. All content on this website is provided by and is the sole responsibility of the person from which such content originated, and such content does not necessarily reflect the opinions of the University administration or the Board of Trustees.

WHAT WE WILL LEARN
How to change the view of an element in real time in the browser, even though we don't have access rights to modifying the file.
Modifying the DOM of an Element in Real Time

Go to http://www-scf.usc.edu/~goei/571-firebug/lesson2.html

Method 1: Using the HTML View & Style Pane

1) **Launch Firebug** and under the HTML View inspect the body element.
   - Easy way to do this: F12, click on the body tag. It is highlighted.

1a) If we hover over: `font: 13px/1.231 Arial;`

    Hovering over the element does 2 things:
    1) shows an info box about the property (if available)
    2) Clicking on `x` will hide the selected property.
Modifying the DOM of an Element in Real Time

Go to http://www-scf.usc.edu/~goei/571-firebug/lesson2.html

Method 1: Using the HTML View & Style Pane

2) Click on the braces { } of the CSS element declaration. Firebug will allow you to add a new css property.

3) Add the following properties:

```
color: white;
background: black;
```

Result... but we are not done yet

The University of Southern California does not screen or control the content on this website and thus does not guarantee the accuracy, integrity, or quality of such content. All content on this website is provided by and is the sole responsibility of the person from which such content originated, and such content does not necessarily reflect the opinions of the University administration or the Board of Trustees.
Go to http://www-scf.usc.edu/~goei/571-firebug/lesson2.html

Method 1: Using the HTML View & Style Pane

4) Inspect the paragraph element, I am a paragraph element with black text and a white background

5) Double click on the innerText of the element, and modify accordingly.
Modifying the DOM of an Element in Real Time

Go to http://www-scf.usc.edu/~goei/571-firebug/lesson2.html

Method 2: Using the Interactive JavaScript Console

1) Click on Console.
Modifying the DOM of an Element in Real Time

Method 2: Using the Interactive JavaScript Console

CONSOLE PANEL

1. The Interactive JavaScript Console. All console messages (console.log), output of execution is displayed here.
2. Write commands here. As long the JavaScript is valid, any command here will be executed in real time.
3. Filtering. We can filter messages by their type. Clicking will cause the console to break on all errors.

MODIFYING THE DOM OF AN ELEMENT IN REAL TIME

2013 (C) Harman Goei
Modifying the DOM of an Element in Real Time

Method 2: Using the Interactive JavaScript Console

1. Execute the following JavaScript line:
   (To execute JavaScript code, type the line starting at >>> and hit Enter)
   `document.body.style.color = "white";`

   Neat! Firebug has autocomplete when typing in JavaScript commands

   Result:

   >>> document.body.style.color = "white"
   "white"

   Console

   Browser view

2. Execute the following lines of JavaScript:
   `document.body.style.backgroundColor = "black";`
   `document.body.children[0].innerHTML = "I am a paragraph with white text and a black background";`

   Result:

   I am a paragraph element with white and a black background.

   The University of Southern California does not screen or control the content on this website and thus does not guarantee the accuracy, integrity, or quality of such content. All content on this website is provided by and is the sole responsibility of the person from which such content originated, and such content does not necessarily reflect the opinions of the University administration or the Board of Trustees
Modifying the DOM of an Element in Real Time

TUTORIAL OBJECTIVES
Transform this page:  http://www-scf.usc.edu/~goei/571-firebug/lesson2.html

HOW WE ACCOMPLISHED THE OBJECTIVE WITH FIREBUG WITH 2 METHODS:

1. Method 1: Changing HTML/CSS Properties using the HTML Panel and the CSS Pane
   Solution: Use the CSS Pane to change the background to black and the text-color, to white
   Solution: Use the HTML pane to change the text content of the element

2. Method 2: Use the Interactive JavaScript Console
   Solution: Write JavaScript DOM object code in the Console (document.body.style.color....)
Side Note: Executing a Block of JavaScript Code

If you have a block of JavaScript code you would like to execute, such as the following:

```javascript
var array = ["Harman", "Goei", "CSCI", "571"];

for(var i = 0; i < array.length; i++) {
    console.log(array[i]);
}
```

Instead of writing line by line, we can the entire block in the console. Here's how:

1. Press the button in the Console tab.
Side Note: Executing a Block of JavaScript Code

The Console Panel should now be the following:

2. Type the code here.

3. Click Run.
Side Note: Executing a Block of JavaScript Code

Result:

```javascript
var array = ["Harman", "Goei", "CSCI", "571"]
for(var i = 0; i < array.length; i++) {
  console.log(array[i]);
}
```

Note: To go back to single-line mode, click on

```
>>> var array = ["Harman", "Goei", "CSCI", "571"]; for(var i = 0; i < array.
```

WHY IS THIS USEFUL?
To write good JavaScript code, you have to test it frequently. We can write functions, test it, and ensure validity through the console. This practice is known as test-driven development.

MODIFYING THE DOM OF AN ELEMENT IN REAL TIME

2013 (C) Harman Goei
Understanding Behavior of JavaScript Code & Detecting Errors with Firebug

**TUTORIAL OBJECTIVES**

1. Step through the behavior of the JavaScript code
2. Understand what happens in the console when JavaScript hits an error

We will be analyzing the following JavaScript:

```javascript
var array = [1,2,3,4,5,6,7,8, "9"];

for(var i = 0; i < array.length; i++) {
    array[i] = array[i] + 1;
}

console.log(array);

setTimeout(function() { x=z; }, 5000);
```

**WHAT THE CODE DOES**

1. Given an array, add 1 to each element.
2. In 5 seconds, a function will do an illegal operation in JavaScript.
Understanding Behavior of JavaScript Code & Detecting Errors with Firebug

1) Go to http://www-scf.usc.edu/~goei/571-firebug/lesson3.html

2) Go to the Console. You should see the following:

```javascript
[2, 3, 4, 5, 6, 7, 8, 9, "91"]
```

**QUESTION:** Why did the last element become 91?
We will analyze this by stepping through the code.

3) Click on Script
Firebug - SCRIPT PANEL

1. The Script Panel View
2. Actions when JavaScript has hit a breakpoint
   1. Rerun – Shift + F8
   2. Continue – F8
   3. Step Into – F11
   4. Step Over – F10
   5. Step Out – Shift F11
Stepping Through JavaScript

4) Create a breakpoint on line number 12 to analyze the problem. **Refresh the page.** (To create a breakpoint, click on the line number)

```html
<!doctype html>
<html>
<head>
<title>Lesson 3: Using the JavaScript Debugger &amp; Detecting JavaScript Errors</title>
<script type="text/javascript">
var array = [1,2,3,4,5,6,7,8, "9"];
for(var i = 0; i < array.length; i++) {
array[i] = array[i] + 1;
}
console.log(array);
</script>
</head>
<body>
</body>
</html>
```

5) JavaScript has stopped on the breakpoint.

**Notice:** The page is still loading because the body element is not loaded.

UNDERSTANDING BEHAVIOR OF JAVASCRIPT CODE & DETECTING ERRORS WITH FIREBUG

2015 (C) Harman Goel
When Firebug Hits a Breakpoint, what else can we see?

Global elements & local elements (click on Watch)

The stack (useful for recursive functions) (click on Stack)

Breakpoints (click on Breakpoints)
Analyzing the Array using the Watch Pane

4) Step through 8 times the dynamic execution trace by clicking on Step Into...
5) Step through once. The 8th element should be now “91”

After pressing Step Into 8 times... Stepping in once after 8

6) We can execute JavaScript code while JavaScript is still in the breakpoint.
Hit 

UNDERSTANDING BEHAVIOR OF JAVASCRIPT CODE & DETECTING ERRORS WITH FIREBUG
Analyzing the Array using the Watch Pane

6) Execute the following JavaScript. (after typing, hit enter)

```javascript
typeof array[8]
```

**QUESTION:** Why did the last element become 91?
It's because the last element is a string, hence it will do concatenation.

**WHY IS STEPPING THROUGH JAVASCRIPT USEFUL?**

1) It detects logic errors in JavaScript code
2) Useful when interpreting data via AJAX, as numbers may be interpreted as strings, and when we do number manipulation, it will cause a similar error as was described.

Speaking of errors, what does Firebug do when it executes the following illegal code?

```javascript
setTimeout(function() { x=z; console.log("Hello there!"); }, 5000);
```
Errors in the Console

setTimeout(function() { x=z; console.log("Hello there!"); }, 5000);

How to detect JavaScript Syntax/Undefined errors the easy way:

1. Launch Firebug. Click on Console in the Firebug toolbar.
2. Due to the nature of the top code, the error will not appear until 5 seconds have passed. Then the following should show:

```javascript
setTimeout(function() { x=z; }, 5000);
```

WHY IS THIS USEFUL?

- By default, JavaScript code stops executing from the line an error occurs.
- If we didn't have Firebug, we would expect Hello there! to appear in the console, but it didn't, and we would debug manually by using alert() or document.write()
Detecting Web Performance Using Firebug

TUTORIAL OBJECTIVE

We will use http://www.google.com for this tutorial.

Determine what requests are made when going to http://www.google.com and how fast was each request.
Detecting Web Performance Using Firebug

1. Go to http://www.google.com
2. Launch Firebug and click on Net
3. Refresh the page.

1. The requests made
2. Filtering
After refreshing, something like the above should appear.

Here’s what we can say about the performance of http://www.google.com:

1. There were 15 requests, 273.5 KB in total size, 243.3 KB from cache.
2. The remote IP is 74.125.239.18: 443 and 74.125.224.162:443 for one req.
3. The status of each request, and what type it was
4. The longest request is 297 ms.
5. We also know the timeline of each request
6. The legend on the right indicates what each request was doing
**Analyzing a Request using Firebug**

1) Expand the request by clicking on +

<table>
<thead>
<tr>
<th>Request URL</th>
<th>Status Code</th>
<th>Server</th>
<th>Size</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET j...</td>
<td>304 Not Modified</td>
<td>ssl.gstatic.com</td>
<td>14.8 KB</td>
<td>74.125.239.15:443</td>
</tr>
<tr>
<td>GET ch...</td>
<td>304 Not Modified</td>
<td>google.com</td>
<td>1.8 KB</td>
<td>74.125.239.18:443</td>
</tr>
<tr>
<td>GET log...</td>
<td>304 Not Modified</td>
<td>google.com</td>
<td>6.8 KB</td>
<td>74.125.239.18:443</td>
</tr>
<tr>
<td>GET re-...</td>
<td>304 Not Modified</td>
<td>google.com</td>
<td>170.4 KB</td>
<td>74.125.239.18:443</td>
</tr>
</tbody>
</table>

**TIP:** Hovering over an image will show an info box of the image.
Analyzing a Request using Firebug

2) We can see the details of the request in the Headers Tab

<table>
<thead>
<tr>
<th>Headers</th>
<th>Cache</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Response Headers</strong></td>
<td></td>
</tr>
<tr>
<td>Age</td>
<td>572849</td>
</tr>
<tr>
<td>Date</td>
<td>Wed, 10 Oct 2012 14:27:22 GMT</td>
</tr>
<tr>
<td>Expires</td>
<td>Thu, 10 Oct 2013 14:27:22 GMT</td>
</tr>
<tr>
<td>Server</td>
<td>GFE/2.0</td>
</tr>
<tr>
<td>X-Firefox-Spyd</td>
<td>3</td>
</tr>
<tr>
<td><strong>Request Headers</strong></td>
<td>view source</td>
</tr>
<tr>
<td>Accept</td>
<td>image/png, image/*;q=0.8, <em>/</em>;q=0.5</td>
</tr>
<tr>
<td>Accept-Encoding</td>
<td>gzip, deflate</td>
</tr>
<tr>
<td>Accept-Language</td>
<td>en-US, en;q=0.5</td>
</tr>
<tr>
<td>Cache-Control</td>
<td>max-age=0</td>
</tr>
<tr>
<td>Connection</td>
<td>keep-alive</td>
</tr>
<tr>
<td>Host</td>
<td>ssl.gstatic.com</td>
</tr>
<tr>
<td>If-Modified-Since</td>
<td>Thu, 29 Mar 2012 23:53:57 GMT</td>
</tr>
<tr>
<td>Referer</td>
<td><a href="https://www.google.com/">https://www.google.com/</a></td>
</tr>
<tr>
<td>User-Agent</td>
<td>Mozilla/5.0 (Windows NT 6.1; WOW64; rv:16.0) Gecko/20100101 Firefox/16.0</td>
</tr>
<tr>
<td><strong>Response Headers From Cache</strong></td>
<td></td>
</tr>
<tr>
<td>Age</td>
<td>572849</td>
</tr>
<tr>
<td>Cache-Control</td>
<td>public, max-age=31536000</td>
</tr>
<tr>
<td>Content-Length</td>
<td>15130</td>
</tr>
<tr>
<td>Content-Type</td>
<td>image/png</td>
</tr>
<tr>
<td>Date</td>
<td>Wed, 10 Oct 2012 14:27:22 GMT</td>
</tr>
<tr>
<td>Expires</td>
<td>Thu, 10 Oct 2013 14:27:22 GMT</td>
</tr>
<tr>
<td>Last-Modified</td>
<td>Thu, 29 Mar 2012 23:53:57 GMT</td>
</tr>
</tbody>
</table>
Analyzing a Request using Firebug

3) We can see what was loaded in the cache from the cache tab

<table>
<thead>
<tr>
<th>Headers</th>
<th>Cache</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Size</td>
<td>15130</td>
</tr>
<tr>
<td>Device</td>
<td>disk</td>
</tr>
<tr>
<td>Expires</td>
<td>Thu Oct 10 2013 07:27:22 GMT-0700 (Pacific Daylight Time)</td>
</tr>
<tr>
<td>Fetch Count</td>
<td>77</td>
</tr>
<tr>
<td>Last Fetched</td>
<td>Tue Oct 16 2012 22:34:52 GMT-0700 (Pacific Daylight Time)</td>
</tr>
<tr>
<td>Last Modified</td>
<td>Tue Oct 16 2012 22:34:52 GMT-0700 (Pacific Daylight Time)</td>
</tr>
</tbody>
</table>

WHY IS ANALYZING PERFORMANCE WITH FIREBUG USEFUL?

We can detect how large a request is, and where the website is really slow - which is how we can improve performance for a site.
Tracking Cookies and Sessions with Firebug

TUTORIAL OBJECTIVE
Track the cookies and sessions created by a website.

1) We will use http://www.piazza.com for this example.

2) Launch Firebug and click on the cookies panel.
## Tracking Cookies and Sessions with Firebug

<table>
<thead>
<tr>
<th>Cookies</th>
<th>Filter</th>
<th>Default (Accept cookies)</th>
</tr>
</thead>
<tbody>
<tr>
<td>last_piazz_user</td>
<td>h6f6kdr4ceu5ca</td>
<td>piazza.com</td>
</tr>
<tr>
<td>__utmld</td>
<td>1</td>
<td>.piazza.com</td>
</tr>
<tr>
<td>piazza_session</td>
<td>&quot;FLKxGIFDLLHxxLxwDK...&amp;x%S5{yl~w%GvX(Hvt50&quot;</td>
<td>piazza.com</td>
</tr>
<tr>
<td>__utma</td>
<td>231435806.1991925790.13...350019821.1350452979.17</td>
<td>.piazza.com</td>
</tr>
<tr>
<td>__utmmb</td>
<td>231435806.9.10.1350452979</td>
<td>.piazza.com</td>
</tr>
<tr>
<td>__utmcc</td>
<td>231435806</td>
<td>.piazza.com</td>
</tr>
<tr>
<td>__utmzc</td>
<td>231435806.1349420816.1...=(direct)&amp;utmcmd=(none)</td>
<td>.piazza.com</td>
</tr>
<tr>
<td>anonymize</td>
<td>disabled</td>
<td>.piazza.com</td>
</tr>
</tbody>
</table>

1. Name and Expand button. 
   Expanding the cookie only reveals the full value of the value attribute.

2. Value of the cookie
3. Domain of the cookie
4. The size
5. The path
6. When it expires / or whether it is a session
Sources


- **Firebug** - [https://getfirebug.com/](https://getfirebug.com/) (Images, and Firebug Console)

- Tutorials were made from scratch, based on my real-life scenarios as a web developer:
  - Tutorial 1: **Client**: I really like this button from http://www.google.com. I want that same button on my website.
  - Tutorial 2: **Client**: Can we compare how the website looks with white text and black background, black background & white text?
  - Tutorial 3: When I was developing a canvas application for USC Viterbi, I was adding numbers like the following: \(2 + 2\). I expected 4, but the result turned out to be 22.
  - Tutorial 4: When I had the lecture in CSCI 571 about Web Performance, I played with Firebug
  - Tutorial 5: Internship made me deal with cookies/sessions