**Client-Side Debugging**

There are many features provided by both Google Chrome and Mozilla Firefox Developer Tools. The three features I decided to write about are Experimenting with HTML & DOM changes, Debugging JavaScript, and Extending Developer Tools by writing Add-ons.

Experimenting with HTML & DOM changes can be achieved easily by using the Elements Tab within the Developer Tools Panel. The elements tab displays the raw CSS AND HTML code of the webpage. When you click on a specific element within that page the code will highlight within the Elements Tab, so you can view the current code. A cool feature with the Elements Tab is the ability to alter the code and view the changes live. This is a great way to stage changes. For example, let's say you have a <div> within a live web page. The Elements tab provides the ability to make changes to the CSS and not affect all users, as you are making the changes locally on your own machine. This can come in handy with staging changes for approvals or opinions.

Debugging JavaScript is one of the most popular features of Google Chrome & Mozilla Firefox Developer Tools. In the Sources tab, there is a Sources panel user interface with three parts: File Navigator, Code Editor, and JavaScript Debugging pane. The File Navigator lists ever file that the page request. The Code Editor displays the contents of that file. The JavaScript Debugging pan, provides numerous tools for inspecting the page's JavaScript. Some of the most popular tools include: breakpoints, step through functionality, and variables value display. Breakpoints allows you to pause the execution of code at a specific spot. Step through functionality, allows you to start from a breakpoint, then execute the code line by line so you can view what happens as it occur. There are a couple of ways you can view variable values as you step through code: The first method is the Scope pane. This shows you what local and global variables are currently defined along with the value. A useful feature is the ability to edit the variable value within this pane, allowing you to perform on the fly debugging. The second method is to use Watch Expressions, which lets you monitor the values of variables over time. You can also store valid JavaScript expressions.

Lastly, one of my favorite features with both Developer Tools is the ability to Extend DevTools by writing Add-ons. If there is some functionality that is not already built into Developer Tools, a developer can specify the panel's content and behavior using HTML, CSS, and JavaScript. In Chrome DevTools, there are two basic ways to build a custom solution. The first is DevTools Extension, which is a Chrome extension that plugs into the DevTools to add functionality and extend its user interface. The second is Debugging Protocol Client, which is a 3rd party that using the Chrome Remote Debugging Protocol to plug into the low-level debugging support in Chrome. In Firefox, there is a prerequisite to install the SDK (software development kit) first, then follow a tutorial to set it up.

**References**

Basques, Kayce; Smith, Andi; Kearney, Meggin; Bakaus, Paul. *Tools for Web Developers*  <https://developers.google.com/web/tools/chrome-devtools/console/>

Mozilla and individual contributors. *Firefox Developer Tools;* <https://developer.mozilla.org/en-US/docs/Tools/Adding_a_panel_to_the_toolbox>