**Retrieving Selector Identification from DOM Elements**

Table of Contents

[Introduction 1](#_Toc435520313)

[Required Software 1](#_Toc435520314)

[Recommended Software 1](#_Toc435520315)

[Order of Preference of selecting elements 2](#_Toc435520316)

[Using the Element Label 2](#_Toc435520317)

[Using the Element ID 2](#_Toc435520318)

[Using the Element Name 3](#_Toc435520319)

[Using the CSS Selector from Chrome Development tools 4](#_Toc435520320)

[Using the Xpath from Chrome Development tools 5](#_Toc435520321)

[Using the CSS Selector Tester 5](#_Toc435520322)

[CSS from Firebug 6](#_Toc435520323)

[Xpath from Firebug 7](#_Toc435520324)

[Other Tricks 7](#_Toc435520325)

[Selector.yml file 8](#_Toc435520326)

# Introduction

Café scripts uses selectors to reference DCOM elements on a web page, these references will be used to either to click or validate web content. There are several ways to reference a DOM element, and use in in a café script. The first way is by name, the other ways require development tools to extract either the css or the xpath selector value.

## Required Software

Chrome browser with CSS Selector Tester,

Firefox Browser with Firebug Add-on 2.0.13

## Recommended Software

Notepad++

# Order of Preference of selecting elements

Sometimes selecting an element from a page is as obvious as using the name, other times a little digging is needed, and other time a lot of digging is required. Here is a list of procedures listed from easy to hard on how to identify buttons, links, textboxes, and other web objects.

1. Element Label
2. Element ID
3. The Name Attribute

* CSS from Chrome
* Xpath from Chrome
* CSS Selector tool
* CSS from Firebug
* Xpath from Firefox Firebug

1. Other Tricks

## Using the Element Label

Using the label of the selector to reference the element, to reference the element, just type of copy and name into the café script. For example, “Upcoming Events”, and “Show Items” buttons.

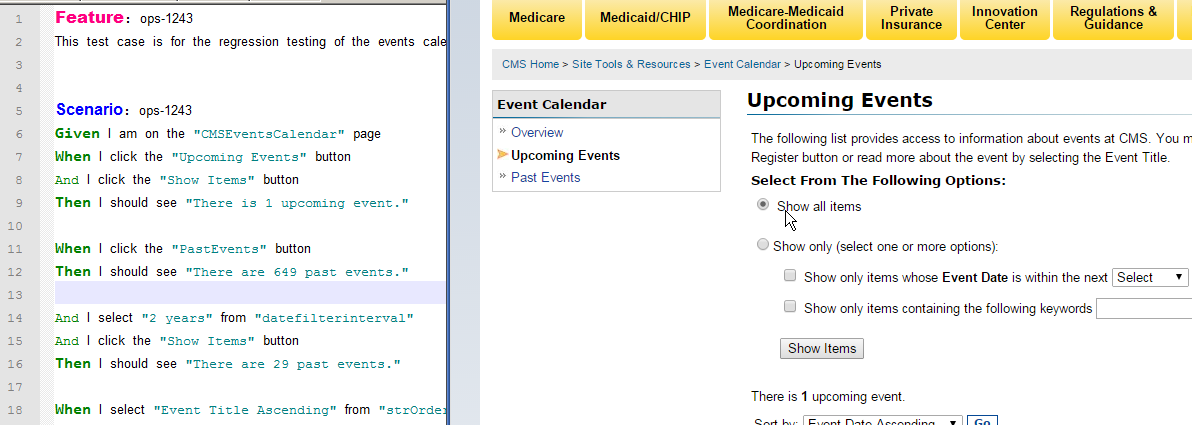


Figure 1 Cafe Script using Element Label

## Using the Element ID

When the easiest form of identification is not working, the label, then the Element ID can be used. This identification method, like the lablel, requires no entry in the selection file. Copy and paste the ID information right into the café script. In this example the Apply button ID is *id="LNTRecordApply", and to use this in a script state When I click the “*LNTRecordApply" button

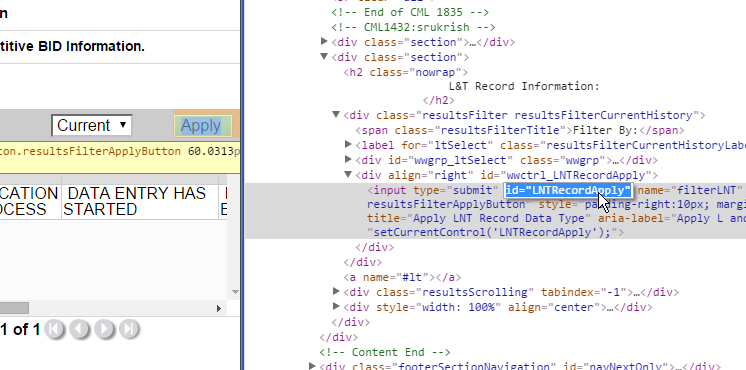


Figure 2 Selection by using ID

## Using the Element Name

When the label or ID fails to identify the selector, try using the next method using the attribute NAME. This method requires an entry in the selector.yml file. To use an attribute encase the attribute in Brackets ‘[]’ with single quotes around the attribute name add a prefix of css and add to the selector.yml file . There are several attributes that can be used, such as title, value, style, etc. but the attribute needs to make the selector unique, such as NAME. The syntax is “css: [attributeName=’value’]” be sure to but the value in single quotes, double quotes won’t work, and won’t through an error message. So in the example we can do :

“css: [title=’Apply LNT Record Data Type’]”

“css:[name=’filterLNT’]”

Either example will work, just make sure it uniquely ids the selector. Combing two attributes such as title and name, requires further understand of using attributes

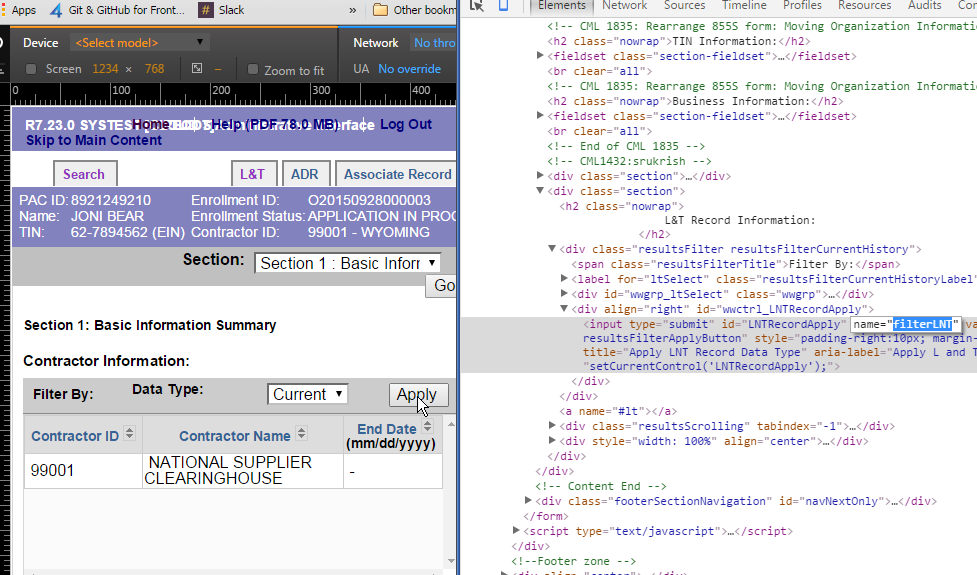


Figure 3 Selection using Attributes

## Using the CSS Selector from Chrome Development tools

The second technique to reference a page Object would be to use the chrome development tools, and to retract the CSS identification value. Right click on the object then select “Inspect Element”. After the right clicking a Development Tool window will be display and the element will be highlighted.

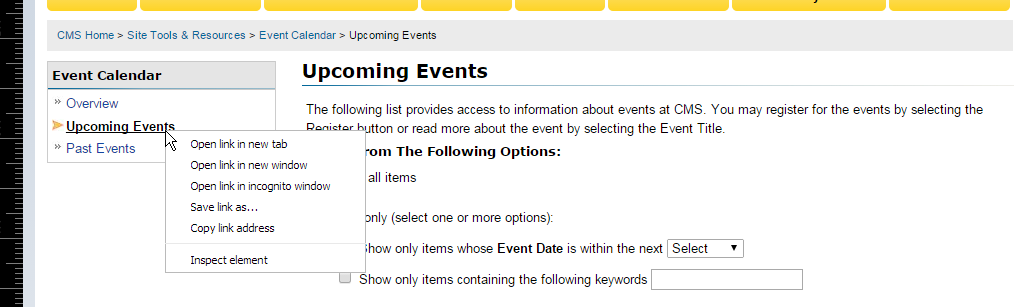


Figure 4 Inspecting Element

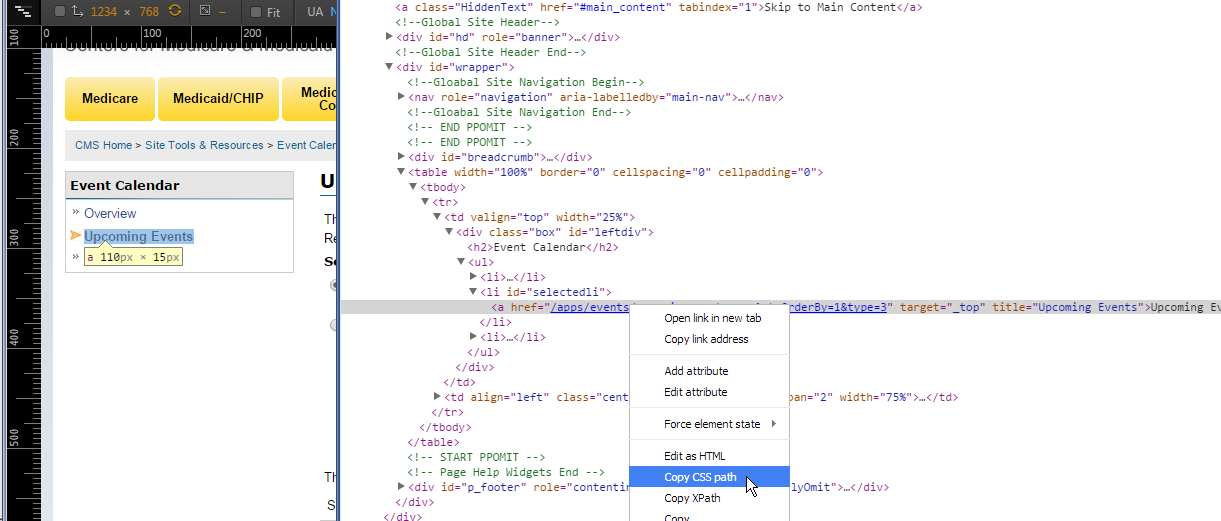


Figure 5 Copying CSS Data

Once the CSS path is copied, it can labled and made into a selector in the selector.yml file. For the above example a entry in the selector.yml file would be

*MyNewSelector: “css: #selectedli > a”*

*MyNewSelector* is the name of the new selector, the name can consist of letter, and number, can’t start with a number, and would be suggested that no spaces were used. Next comes the colon, space, quote and word css and another colon, all required lowercase, and spaces after first colon. After the selector is created in the yml file it can be used in the café script such as this

When I click the "*MyNewSelector*" button

## Using the Xpath from Chrome Development tools

Sometimes the CSS selector fails to identity the DOC object, so the next technique is using the Xpath. The syntax for the Xpath is

*MyNewSelector: “xpath:* //\*[@id=’selectedli’]/a*”*

To copy the xpath information is identical to method used to get the css, right click select inspect element, then select Copy Xpath.

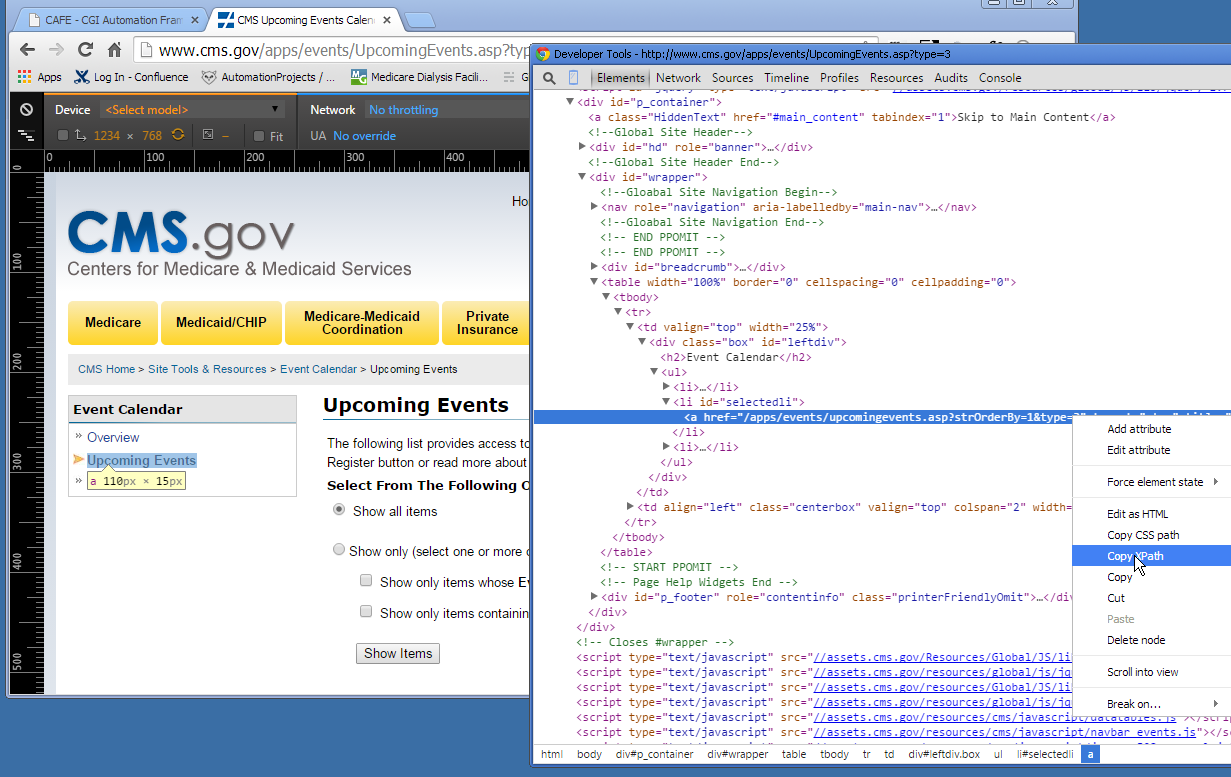


Figure 6 Copying the Xpath Data

Before using the xpah, the double quotes have to be changed to single quotes, because there are double quotes surrounding the entire xpath string. Such as in

//\*[@id="selectedli"]/a (double quotes) to //\*[@id=’selectedli’]/a (single quotes)

## Using the CSS Selector Tester

When all techniques fail, label, css, xpath, the last resort is to use the css selector add-on tool for chrome. This is an addon that can be access from chrome.

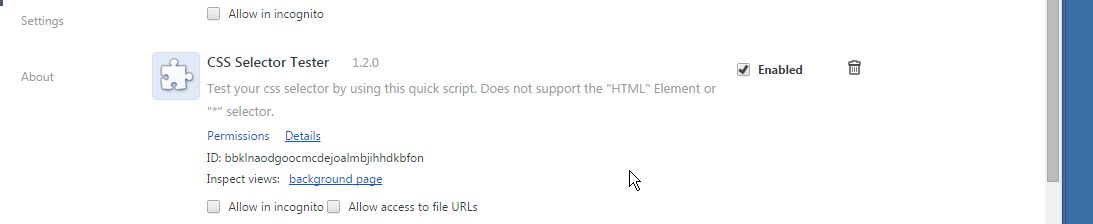


Figure 7 CSS Selector Tester Chrome Tool

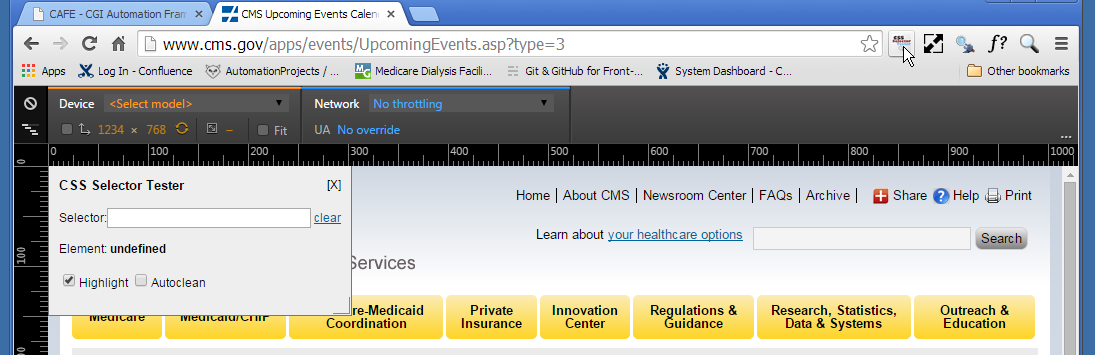


Figure 8 Selector Tool Usage

To use the CSS Selector Tester, right click on the object and inspect the element , right click and copy the CSS path, then paste the CSS information into the Selector text box

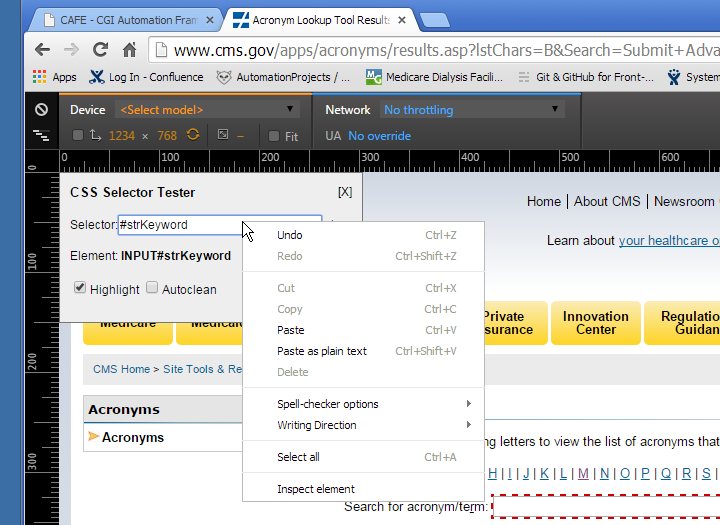


Figure 9 Pasting CSS Info into Selector Tool

After pasting css information press enter, and the tool will generate a reference for the element, I this case the reference is INPUT#strKeyboard. This reference is then place in the selector.yml file as a CSS reference such as

*MyNewSelector: “css:* INPUT#strKeyboard”

## CSS from Firebug

When all other methods fail to id the object, firebug on firefox, can sometimes help. This method require an selector.yml entry, and the css that firebug generates is much long text then the xpath. Here is the example:

Right click on the object and select Inspect Element with Firebug

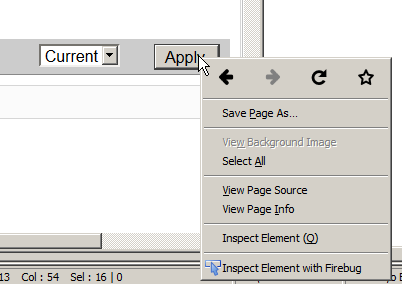


Figure 10 Using Firebug

Then select the “Copy Css Path”

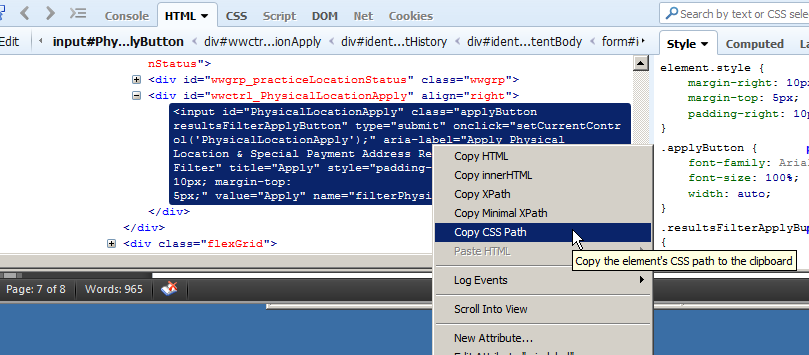


Figure 11 Inspecting Element using Firebug

For this object the css from firebug is quite lengthy

*html body div#main div.bodyContent form#identifyingInfoSummary\_action div#identifyingInfoSummary\_action\_.contentBody div#identifyingInfoSummary\_action\_.resultsFilter.resultsFilterCurrentHistory div#wwctrl\_PhysicalLocationApply input#PhysicalLocationApply.applyButton.resultsFilterApplyButton*

## Xpath from Firebug

The xpath identification values generated from firebug xpath, is not as lengthy and is easier to work with then the css values. Same steps apply but click on *Copy Xpath.* The xpath values generated from the

*Apply* button in the xpath example, are much smaller than the css information

*/html/body/div/div[5]/form/div[1]/div[4]/div[3]/input*

To use in the selector file would be

SelectorName: “xpath: /html/body/div/div[5]/form/div[1]/div[4]/div[3]/input”

## Other Tricks

Sometimes when all else fails, some tricks can be used.

* Removing or adding the ‘/a’ in the xpath such as *MyNewSelector: “xpath:* //\*[@id=’selectedli’]/a*”.* if there is an ‘/a” remove it, if not add it .
* Getting the css or the xapth from the parent object. In chrome the ticks opens and closed the parent/child objects,, the parent is the id="wwgrp\_npiSelect"\_
* 
* Figure 12 Parent Child Object relationship
* Sometimes objects are in table and changing the div index number in the xpath, will find the right selector, for example “xpath: /html/body/div/div[5]/form/div[1]/div[4]/div[3]/input”, look at the page and determine how many ‘Apply’ buttons are present, then start with the right most number, ‘3’ in this case, and change according, but remember computers count starting at 0.

# Selector.yml file

The selector.yml file has a tendency to get large depending on the amount of css, and xpath selectors that are needed in the script. Blocking the selectors together according to function and using comment to descript where the selectors are used, will lessen the chaos of a large selector file.

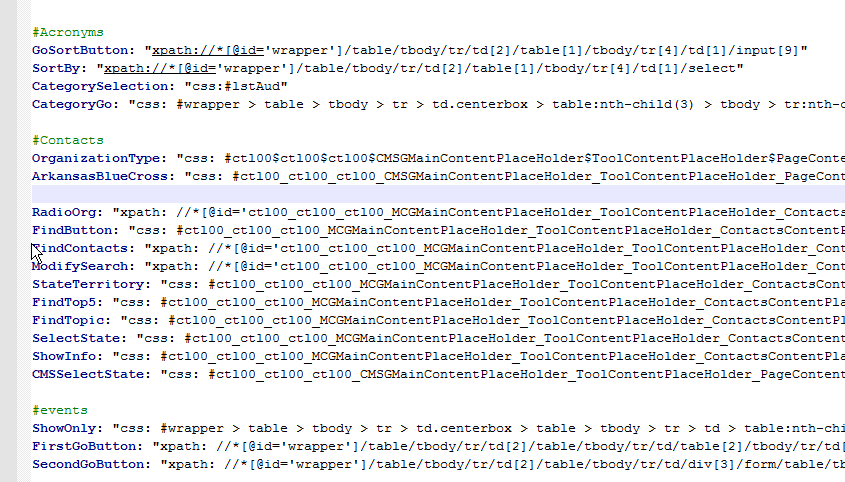


Figure 13 Sample Selector.yml file