

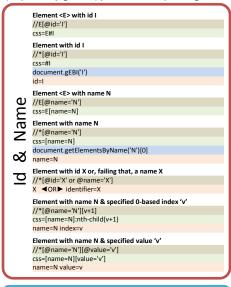
Element <E> by relative reference

css=F document.gEBTN('E')[0] Second <E> element anywhere on page xpath=(//E)[2] document.gEBTN('E')[1] Image element //img css=img document.images[0] Flement <F> with attribute A //E[@A] dom=for each (e in document.gEBTN('E')) if (e.A) e 0 Element <E> with attribute A containing text 't' exactly //E[@A='t'] css=E[A='t'] @ Element <E> with attribute A containing text 't' //E[contains(@A.'t')] css=E[A*='t'] @ Element <E> whose attribute A begins with 't' //E[starts-with(@A, 't')] css=F[A^='t'] @ Element <E> whose attribute A ends with 't' //E[ends-with(@A, 't')] **▼**◀OR //E[substring(@A, string-length(@A) - string-length('t')+1)='t'] css=E[AS='t'] @ Element <E> with attribute A containing word 'w' //E[contains(concat('⊕', @A, '⊕'), '⊕w⊕') Element <E> with attribute A matching regex 'r' //E[matches(@A, 'r')] 🖎 Element <E1> with id I1 or element <E2> with id I2 //E1[@id=I1] | //E2[@id=I2] css=E1#I1,E2#I2 Element <E1> with id I1 or id I2 //E1[@id=I1 or @id=I2] css=E1#I1,E1#I2

Attribute A of element <E> //E/@A 🖾 {Se: //E@A } {Se: css=F@A } 0 document.gEBTN('E')[0].getAttribute('A') {Se: document.gEBTN('E')[0]@A } Attribute A of any element //*/@A 🖾 {Se://*@A } Attribute A1 of element <E> where attribute A2 is 't' exactly //E[@A2='t']/@A1 🗭 {Se: //E[@A2='t']@A1 } {Se: css=E[A2='t']@A1 } Attribute A of element <E> where A contains 't' //E[contains(@A,'t')]/@A ☒ {Se: //E[contains(@A,'t')]@A }

XPATH • CSS • DOM • SELENIUM Rosetta Stone and Cookbook

Sprinkled with Selenium usage tips, this is both a general-purpose set of recipes for each technology as well as a cross-reference to map from one to another. The validation suite for this reference chart (http://bit.ly/gTd5oc) provides example usage for each recipe supported by Selenium (the majority of them).



Element <E> is explicitly in language L or subcode //E[@lang='L' or starts-with(@lang, concat('L', '-'))] css=E[lang|=L] Element <E> is in language L or subcode (possibly inherited) ᆲ css=E:lang(L) ∞ Element with a class C //*[contains(concat('⊕', @class, '⊕'), '⊕C⊕')] document.getElementsByClassName('C')[0] Element <E> with a class C //E[contains(concat('⊕', @class, '⊕'), '⊕C⊕')]

```
Element containing text 't' exactly
      //*[.='t']
      Element <E> containing text 't'
      //E[contains(text(),'t')]
      css=E:contains('t')
Link
      Link element
      //a
      css=a
      document.links[0]
      <a> containing text 't' exactly
Ø
      //a[.='t']
      link=t
      <a> containing text 't'
      //a[contains(text(),'t')]
css=a:contains('t') 4
      <a> with target link 'url'
      //a[@href='url']
      css=a[href='url']
      Link URL labeled with text 't' exactly
```

First child of element <F> //E/*[1]
css=E > *:first-child {Se: css=E > * } document.gEBTN('E')[0].firstChild 6 First <E> child css=E:first-of-type 🔀 {Se: css=E } document.getEBTN('E')[0] Last child of element E //E/*[last()] css=E *:last-child document.gEBTN('E')[0].lastChild 6 Last <E> child //E[last()] css=E:last-of-type document.gEBTN(E)[document.gEBTN(E).length-1] Second <F> child //E[2] ◀OR▶ //E/following-sibling::E css=E:nth-of-type(2)
document.getEBTN('E')[1] Second child that is an <E> element //*[2][name()='E'] css=E:nth-child(2) Second-to-last <E> child //F[last()-1] css=E:nth-last-of-type(2) document.gEBTN(E)[document.gEBTN(E).length-2] Second-to-last child that is an <E> element //*[last()-1][name()='E'] css=E:nth-last-child(2) Element <E1> with only <E2> children //E1/[E2 and not(*[not(self::E2)])] Parent of element <F> document.gEBTN('E')[0].parentNode Descendant <E> of element with id I using specific path //*[@id='l']/ . . ./. . ./E document.gEBI('I')...gEBTN('E')[0] Descendant <E> of element with id I using unspecified path //*[@id='I']//E css=#LF document.gEBI('I').gEBTN('E')[0] Element <E> with no children //E[count(*)=0] css=F:empty Element <E> with an only child //E[count(*)=1] Element <E> that is an only child //E[count(preceding-sibling::*)+count(following-sibling::*)=0] Element <E> with no <E> siblings //E[count(../F) = 1] css=E:only-of-type Every Nth element starting with the (M+1)th //E[position() mod N = M + 1]css=F:nth-child(Nn + M)

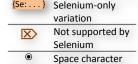
Element <E1> following some sibling <E2> //E2/following-sibling::E1 css=E2 ~ E1 Element <E1> immediately following sibling <E2> //E2/following-sibling::*[1][name()='E1'] Element <E1> following sibling <E2> with one intermediary //E2/following-sibling::*[2][name()='E1'] css=E2 + * + E1 Sibling element immediately following <E> plin //E/following-sibling::* css=F + 3 둜 document.gEBTN('E')[0].nextSibling Element <E1> preceding some sibling <E2> //E2/preceding-sibling::E1 Element <E1> immediately preceding sibling <E2> //E2/preceding-sibling::*[1][name()='E1'] Element <E1> preceding sibling <E2> with one intermediary //E2/preceding-sibling::*[2][name()='E1'] Sibling element immediately preceding <E> //E/preceding-sibling::*[1] document.gEBTN('E2')[0].previousSibling 6

Cell by row and column (e.g. 3rd row, 2nd column) //*[@id='TestTable']//tr[3]//td[2] {Se: //*[@id='TestTable'].2.1 } css=#TestTable tr:nth-child(3) td:nth-child(2) {Se: css=#TestTable.2.1 } document.gEBI('TestTable').gEBTN('tr')[2].gEBTN('td')[1] {Se: document.gEBI('TestTable').2.1 } ᅙ Cell immediately following cell containing 't' exactly //td[preceding-sibling::td='t'] Cell immediately following cell containing 't' //td[preceding-sibling::td[contains(.,'t')]] css=td:contains('t') ~ td @



• DOM has limited capability with a simple 'document...' expression; however, arbitrary JavaScript code may be used as shown in this example

- CSS does not support qualifying elements with the style attribute, as in div[style*='border-width'].
- Selenium uses a special syntax for returning attributes; normal XPath, CSS, and DOM syntax will fail.
- **②** CSS: The CSS2 contains function is *not in CSS3*; however, Selenium supports the superset of CSS1, 2, and 3.
- **5** DOM: firstChild, lastChild, nextSibling, and previousSibling are problematic with mixed content; they will point to empty text nodes rather than desired elements depending on whitespace in web page source.



expression CSS3 or XPath 2.0 DOM abbreviations:

XPath CSS DOM Selenium gEBI getElementBvId gEBTN getElementsByTagName

Indexing (all): XPath and CSS use 1-based indexing; DOM and Selenium's table syntax use 0-based indexing.

Cardinality (Selenium): XPath and CSS may specify a node set or a single node; DOM must specify a single node. When a node set is specified, Selenium returns just the first node. Content (XPath): Generally should use normalize-space() when operating on display text.

