**Interface WebElement**

* **All Superinterfaces:**

[SearchContext](https://www.selenium.dev/selenium/docs/api/java/org/openqa/selenium/SearchContext.html), [TakesScreenshot](https://www.selenium.dev/selenium/docs/api/java/org/openqa/selenium/TakesScreenshot.html)

**All Known Implementing Classes:**

[RemoteWebElement](https://www.selenium.dev/selenium/docs/api/java/org/openqa/selenium/remote/RemoteWebElement.html)

public interface **WebElement**

extends [SearchContext](https://www.selenium.dev/selenium/docs/api/java/org/openqa/selenium/SearchContext.html), [TakesScreenshot](https://www.selenium.dev/selenium/docs/api/java/org/openqa/selenium/TakesScreenshot.html)

Represents an HTML element. Generally, all interesting operations to do with interacting with a page will be performed through this interface.

All method calls will do a freshness check to ensure that the element reference is still valid. This essentially determines whether or not the element is still attached to the DOM. If this test fails, then an [StaleElementReferenceException](https://www.selenium.dev/selenium/docs/api/java/org/openqa/selenium/StaleElementReferenceException.html) is thrown, and all future calls to this instance will fail.

* + ***Method Summary***

|  |  |  |
| --- | --- | --- |
| **Modifier and Type** | **Method** | **Description** |
| void | [**clear**](https://www.selenium.dev/selenium/docs/api/java/org/openqa/selenium/WebElement.html#clear())() | If this element is a form entry element, this will reset its value. |
| void | [**click**](https://www.selenium.dev/selenium/docs/api/java/org/openqa/selenium/WebElement.html#click())() | Click this element. |
| [**WebElement**](https://www.selenium.dev/selenium/docs/api/java/org/openqa/selenium/WebElement.html) | [**findElement**](https://www.selenium.dev/selenium/docs/api/java/org/openqa/selenium/WebElement.html#findElement(org.openqa.selenium.By))​([**By**](https://www.selenium.dev/selenium/docs/api/java/org/openqa/selenium/By.html) by) | Find the first [WebElement](https://www.selenium.dev/selenium/docs/api/java/org/openqa/selenium/WebElement.html) using the given method. |
| java.util.List<[**WebElement**](https://www.selenium.dev/selenium/docs/api/java/org/openqa/selenium/WebElement.html)> | [**findElements**](https://www.selenium.dev/selenium/docs/api/java/org/openqa/selenium/WebElement.html#findElements(org.openqa.selenium.By))​([**By**](https://www.selenium.dev/selenium/docs/api/java/org/openqa/selenium/By.html) by) | Find all elements within the current context using the given mechanism. |
| default java.lang.String | [**getAccessibleName**](https://www.selenium.dev/selenium/docs/api/java/org/openqa/selenium/WebElement.html#getAccessibleName())() | Gets result of a Accessible Name and Description Computation for the Accessible Name of the element. |
| default java.lang.String | [**getAriaRole**](https://www.selenium.dev/selenium/docs/api/java/org/openqa/selenium/WebElement.html#getAriaRole())() | Gets result of computing the WAI-ARIA role of element. |
| java.lang.String | [**getAttribute**](https://www.selenium.dev/selenium/docs/api/java/org/openqa/selenium/WebElement.html#getAttribute(java.lang.String))​(java.lang.String name) | Get the value of the given attribute of the element. |
| java.lang.String | [**getCssValue**](https://www.selenium.dev/selenium/docs/api/java/org/openqa/selenium/WebElement.html#getCssValue(java.lang.String))​(java.lang.String propertyName) | Get the value of a given CSS property. |
| default java.lang.String | [**getDomAttribute**](https://www.selenium.dev/selenium/docs/api/java/org/openqa/selenium/WebElement.html#getDomAttribute(java.lang.String))​(java.lang.String name) | Get the value of the given attribute of the element. |
| default java.lang.String | [**getDomProperty**](https://www.selenium.dev/selenium/docs/api/java/org/openqa/selenium/WebElement.html#getDomProperty(java.lang.String))​(java.lang.String name) | Get the value of the given property of the element. |
| [**Point**](https://www.selenium.dev/selenium/docs/api/java/org/openqa/selenium/Point.html) | [**getLocation**](https://www.selenium.dev/selenium/docs/api/java/org/openqa/selenium/WebElement.html#getLocation())() | Where on the page is the top left-hand corner of the rendered element? |
| [**Rectangle**](https://www.selenium.dev/selenium/docs/api/java/org/openqa/selenium/Rectangle.html) | [**getRect**](https://www.selenium.dev/selenium/docs/api/java/org/openqa/selenium/WebElement.html#getRect())() |  |
| default [**SearchContext**](https://www.selenium.dev/selenium/docs/api/java/org/openqa/selenium/SearchContext.html) | [**getShadowRoot**](https://www.selenium.dev/selenium/docs/api/java/org/openqa/selenium/WebElement.html#getShadowRoot())() |  |
| [**Dimension**](https://www.selenium.dev/selenium/docs/api/java/org/openqa/selenium/Dimension.html) | [**getSize**](https://www.selenium.dev/selenium/docs/api/java/org/openqa/selenium/WebElement.html#getSize())() | What is the width and height of the rendered element? |
| java.lang.String | [**getTagName**](https://www.selenium.dev/selenium/docs/api/java/org/openqa/selenium/WebElement.html#getTagName())() | Get the tag name of this element. |
| java.lang.String | [**getText**](https://www.selenium.dev/selenium/docs/api/java/org/openqa/selenium/WebElement.html#getText())() | Get the visible (i.e. |
| boolean | [**isDisplayed**](https://www.selenium.dev/selenium/docs/api/java/org/openqa/selenium/WebElement.html#isDisplayed())() | Is this element displayed or not? This method avoids the problem of having to parse an element's "style" attribute. |
| boolean | [**isEnabled**](https://www.selenium.dev/selenium/docs/api/java/org/openqa/selenium/WebElement.html#isEnabled())() | Is the element currently enabled or not? This will generally return true for everything but disabled input elements. |
| boolean | [**isSelected**](https://www.selenium.dev/selenium/docs/api/java/org/openqa/selenium/WebElement.html#isSelected())() | Determine whether or not this element is selected or not. |
| void | [**sendKeys**](https://www.selenium.dev/selenium/docs/api/java/org/openqa/selenium/WebElement.html#sendKeys(java.lang.CharSequence...))​(java.lang.CharSequence... keysToSend) | Use this method to simulate typing into an element, which may set its value. |
| void | [**submit**](https://www.selenium.dev/selenium/docs/api/java/org/openqa/selenium/WebElement.html#submit())() | If this current element is a form, or an element within a form, then this will be submitted to the remote server. |

* + - **Methods inherited from interface org.openqa.selenium.**[**TakesScreenshot**](https://www.selenium.dev/selenium/docs/api/java/org/openqa/selenium/TakesScreenshot.html)

[getScreenshotAs](https://www.selenium.dev/selenium/docs/api/java/org/openqa/selenium/TakesScreenshot.html#getScreenshotAs(org.openqa.selenium.OutputType))

* + ***Method Detail***
    - **click**

void click()

Click this element. If this causes a new page to load, you should discard all references to this element and any further operations performed on this element will throw a StaleElementReferenceException.

Note that if click() is done by sending a native event (which is the default on most browsers/platforms) then the method will \_not\_ wait for the next page to load and the caller should verify that themselves.

There are some preconditions for an element to be clicked. The element must be visible and it must have a height and width greater then 0.

**Throws:** [StaleElementReferenceException](https://www.selenium.dev/selenium/docs/api/java/org/openqa/selenium/StaleElementReferenceException.html) - If the element no longer exists as initially defined

* + - **submit**

void submit()

If this current element is a form, or an element within a form, then this will be submitted to the remote server. If this causes the current page to change, then this method will block until the new page is loaded.

**Throws:** [NoSuchElementException](https://www.selenium.dev/selenium/docs/api/java/org/openqa/selenium/NoSuchElementException.html) - If the given element is not within a form

* + - **sendKeys**

void sendKeys​(java.lang.CharSequence... keysToSend)

Use this method to simulate typing into an element, which may set its value.

**Parameters:** keysToSend - character sequence to send to the element

**Throws:** java.lang.IllegalArgumentException - if keysToSend is null

* + - **clear**

void clear()

If this element is a form entry element, this will reset its value.

* + - **getTagName**

java.lang.String getTagName()

Get the tag name of this element. **Not** the value of the name attribute: will return "input" for the element <input name="foo" />.

**Returns:** The tag name of this element.

* + - **getDomProperty**

default java.lang.String getDomProperty​(java.lang.String name)

Get the value of the given property of the element. Will return the current value, even if this has been modified after the page has been loaded.

**Parameters:** name - The name of the property.

**Returns:** The property's current value or null if the value is not set.

* + - **getDomAttribute**

default java.lang.String getDomAttribute​(java.lang.String name)

Get the value of the given attribute of the element.

This method, unlike [getAttribute(String)](https://www.selenium.dev/selenium/docs/api/java/org/openqa/selenium/WebElement.html#getAttribute(java.lang.String)), returns the value of the attribute with the given name but not the property with the same name.

The following are deemed to be "boolean" attributes, and will return either "true" or null:

async, autofocus, autoplay, checked, compact, complete, controls, declare, defaultchecked, defaultselected, defer, disabled, draggable, ended, formnovalidate, hidden, indeterminate, iscontenteditable, ismap, itemscope, loop, multiple, muted, nohref, noresize, noshade, novalidate, nowrap, open, paused, pubdate, readonly, required, reversed, scoped, seamless, seeking, selected, truespeed, willvalidate

**Parameters:** name - The name of the attribute.

**Returns:** The attribute's value or null if the value is not set.

* + - **getAttribute**

java.lang.String getAttribute​(java.lang.String name)

Get the value of the given attribute of the element. Will return the current value, even if this has been modified after the page has been loaded.

More exactly, this method will return the value of the property with the given name, if it exists. If it does not, then the value of the attribute with the given name is returned. If neither exists, null is returned.

The "style" attribute is converted as best can be to a text representation with a trailing semi-colon.

The following are deemed to be "boolean" attributes, and will return either "true" or null:

async, autofocus, autoplay, checked, compact, complete, controls, declare, defaultchecked, defaultselected, defer, disabled, draggable, ended, formnovalidate, hidden, indeterminate, iscontenteditable, ismap, itemscope, loop, multiple, muted, nohref, noresize, noshade, novalidate, nowrap, open, paused, pubdate, readonly, required, reversed, scoped, seamless, seeking, selected, truespeed, willvalidate

Finally, the following commonly mis-capitalized attribute/property names are evaluated as expected:

* + - * If the given name is "class", the "className" property is returned.
      * If the given name is "readonly", the "readOnly" property is returned.

*Note:* The reason for this behavior is that users frequently confuse attributes and properties. If you need to do something more precise, use [getDomAttribute(String)](https://www.selenium.dev/selenium/docs/api/java/org/openqa/selenium/WebElement.html#getDomAttribute(java.lang.String)) or [getDomProperty(String)](https://www.selenium.dev/selenium/docs/api/java/org/openqa/selenium/WebElement.html#getDomProperty(java.lang.String)) to obtain the result you desire.

**Parameters:** name - The name of the attribute.

**Returns:** The attribute/property's current value or null if the value is not set.

* + - **getAriaRole**

default java.lang.String getAriaRole()

Gets result of computing the WAI-ARIA role of element.

See [W3C WebDriver specification](https://www.w3.org/TR/webdriver/#get-computed-role) for more details.

**Returns:** the WAI-ARIA role of the element.

* + - **getAccessibleName**

default java.lang.String getAccessibleName()

Gets result of a Accessible Name and Description Computation for the Accessible Name of the element.

**Returns:** the accessible name of the element.

* + - **isSelected**

boolean isSelected()

Determine whether or not this element is selected or not. This operation only applies to input elements such as checkboxes, options in a select and radio buttons. For more information on which elements this method supports, refer to the [specification](https://w3c.github.io/webdriver/webdriver-spec.html#is-element-selected).

**Returns:** True if the element is currently selected or checked, false otherwise.

* + - **isEnabled**

boolean isEnabled()

Is the element currently enabled or not? This will generally return true for everything but disabled input elements.

**Returns:** True if the element is enabled, false otherwise.

* + - **getText**

java.lang.String getText()

Get the visible (i.e. not hidden by CSS) text of this element, including sub-elements.

**Returns:** The visible text of this element.

* + - **findElements**

java.util.List<[WebElement](https://www.selenium.dev/selenium/docs/api/java/org/openqa/selenium/WebElement.html)> findElements​([By](https://www.selenium.dev/selenium/docs/api/java/org/openqa/selenium/By.html) by)

Find all elements within the current context using the given mechanism. When using xpath be aware that webdriver follows standard conventions: a search prefixed with "//" will search the entire document, not just the children of this current node. Use ".//" to limit your search to the children of this WebElement. This method is affected by the 'implicit wait' times in force at the time of execution. When implicitly waiting, this method will return as soon as there are more than 0 items in the found collection, or will return an empty list if the timeout is reached.

**Specified by:** [findElements](https://www.selenium.dev/selenium/docs/api/java/org/openqa/selenium/SearchContext.html#findElements(org.openqa.selenium.By)) in interface [SearchContext](https://www.selenium.dev/selenium/docs/api/java/org/openqa/selenium/SearchContext.html)

**Parameters:** by - The locating mechanism to use

**Returns:** A list of all [WebElement](https://www.selenium.dev/selenium/docs/api/java/org/openqa/selenium/WebElement.html)s, or an empty list if nothing matches.

* + - **findElement**

[WebElement](https://www.selenium.dev/selenium/docs/api/java/org/openqa/selenium/WebElement.html) findElement​([By](https://www.selenium.dev/selenium/docs/api/java/org/openqa/selenium/By.html) by)

Find the first [WebElement](https://www.selenium.dev/selenium/docs/api/java/org/openqa/selenium/WebElement.html) using the given method. See the note in [findElements(By)](https://www.selenium.dev/selenium/docs/api/java/org/openqa/selenium/WebElement.html#findElements(org.openqa.selenium.By)) about finding via XPath. This method is affected by the 'implicit wait' times in force at the time of execution. The findElement(..) invocation will return a matching row, or try again repeatedly until the configured timeout is reached.

findElement should not be used to look for non-present elements, use [findElements(By)](https://www.selenium.dev/selenium/docs/api/java/org/openqa/selenium/WebElement.html#findElements(org.openqa.selenium.By)) and assert zero length response instead.

**Specified by:** [findElement](https://www.selenium.dev/selenium/docs/api/java/org/openqa/selenium/SearchContext.html#findElement(org.openqa.selenium.By)) in interface [SearchContext](https://www.selenium.dev/selenium/docs/api/java/org/openqa/selenium/SearchContext.html)

**Parameters:** by - The locating mechanism

**Returns:** The first matching element on the current context.

**Throws:** [NoSuchElementException](https://www.selenium.dev/selenium/docs/api/java/org/openqa/selenium/NoSuchElementException.html) - If no matching elements are found

* + - **getShadowRoot**

default [SearchContext](https://www.selenium.dev/selenium/docs/api/java/org/openqa/selenium/SearchContext.html) getShadowRoot()

**Returns:** A representation of an element's shadow root for accessing the shadow DOM of a web component.

**Throws:** [NoSuchShadowRootException](https://www.selenium.dev/selenium/docs/api/java/org/openqa/selenium/NoSuchShadowRootException.html) - If no shadow root is found

* + - **isDisplayed**

boolean isDisplayed()

Is this element displayed or not? This method avoids the problem of having to parse an element's "style" attribute.

**Returns:** Whether or not the element is displayed

* + - **getLocation**

[Point](https://www.selenium.dev/selenium/docs/api/java/org/openqa/selenium/Point.html) getLocation()

Where on the page is the top left-hand corner of the rendered element?

**Returns:** A point, containing the location of the top left-hand corner of the element

* + - **getSize**

[Dimension](https://www.selenium.dev/selenium/docs/api/java/org/openqa/selenium/Dimension.html) getSize()

What is the width and height of the rendered element?

**Returns:** The size of the element on the page.

* + - **getRect**

[Rectangle](https://www.selenium.dev/selenium/docs/api/java/org/openqa/selenium/Rectangle.html) getRect()

**Returns:** The location and size of the rendered element

* + - **getCssValue**

java.lang.String getCssValue​(java.lang.String propertyName)

Get the value of a given CSS property. Color values could be returned as rgba or rgb strings. This depends on whether the browser omits the implicit opacity value or not. For example if the "background-color" property is set as "green" in the HTML source, the returned value could be "rgba(0, 255, 0, 1)" if implicit opacity value is preserved or "rgb(0, 255, 0)" if it is omitted. Note that shorthand CSS properties (e.g. background, font, border, border-top, margin, margin-top, padding, padding-top, list-style, outline, pause, cue) are not returned, in accordance with the [DOM CSS2 specification](http://www.w3.org/TR/DOM-Level-2-Style/css.html#CSS-CSSStyleDeclaration) - you should directly access the longhand properties (e.g. background-color) to access the desired values.

**Parameters:** propertyName - the css property name of the element

**Returns:** The current, computed value of the property.