POWERING POTENTIAL

Selenium

Module 3 – Interacting with Elements

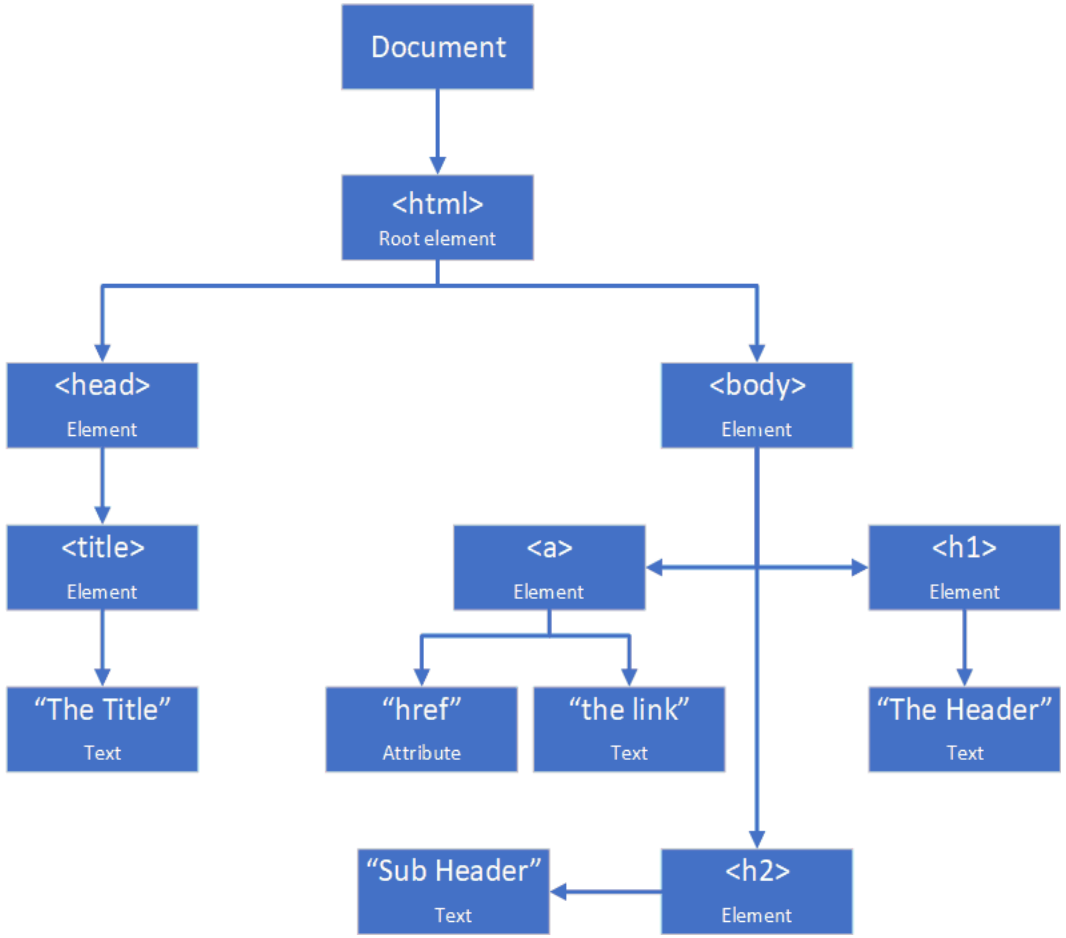
The Document Object model

A browser models a Web page as a tree of objects called a DOM. This allows programs and scripts to dynamically access and update a document's:

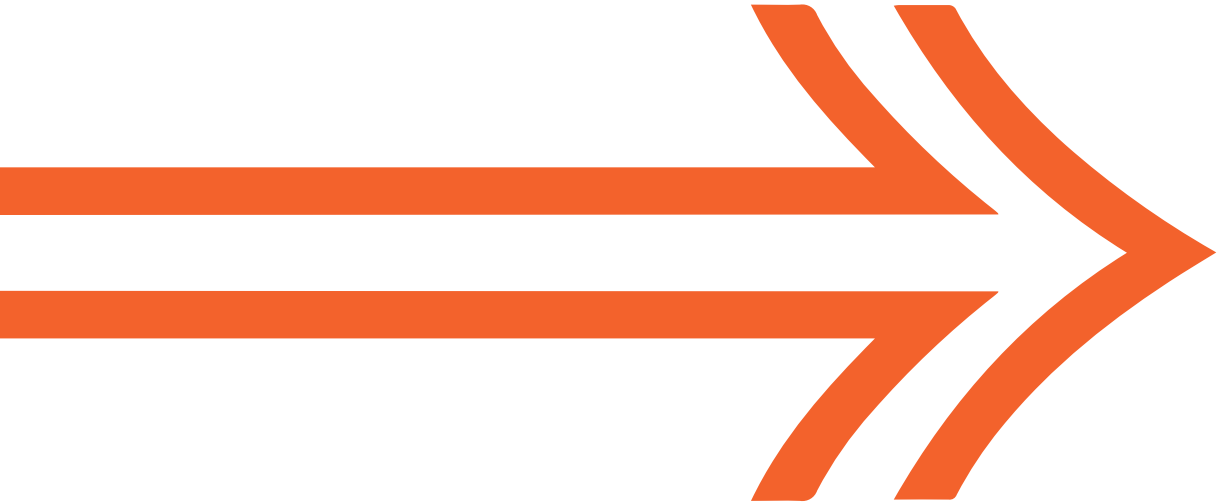
- content
- structure
- style

The DOM defines the following:

- All HTML elements as objects.
- The properties of all HTML elements.
- The methods that can access all HTML elements.
- The events that can affect all HTML elements.



Interacting with Elements



- Each tag on a Web page is an **element within the DOM,** which can be located and interacted with.
- The **By** class is used to select an element.
- The **WebElement** interface has a set of actions which can be performed on every element.
  - Finding elements by their id is typically the best as it will always be unique.

Locating Elements

APPLYING AN ELEMENT SELECTION STRATEGY

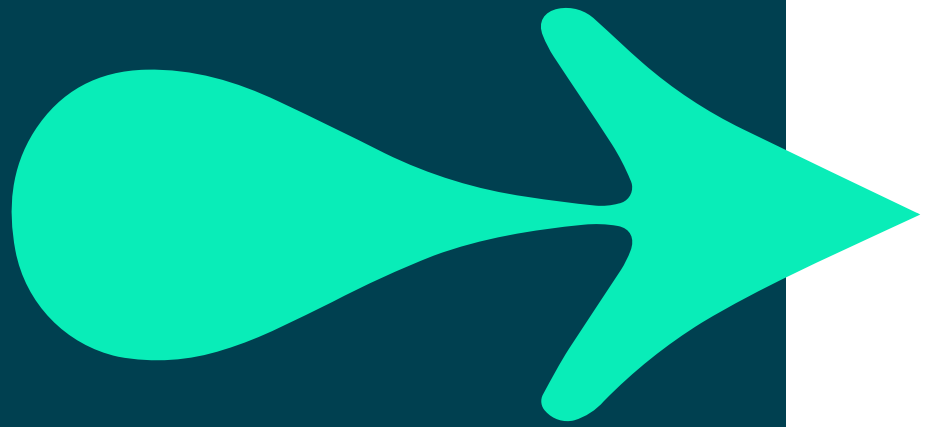
```
By searchBarSelector = By.name("q");
WebElement searchBar = driver.findElement(searchBarSelector);
```

- Selenium offers multiple built-in element selection strategies to find elements on a web page, these are specified on the **By** abstract class which is an API for creating **WebElement** selectors programmatically.

- We use static methods on the **By** interface to create concrete element selectors of various types.

- Once we have created some concrete By implementation using its static methods, we can then use the WebDriver API to find the element or elements by the given selector.

SEARCH CONTEXTS



- You can search for DOM nodes within a selected **WebElement** either via a **ByCssSelector** or the **WebElement** itself.

- A **WebElement** can call the **findElement(By selector)** method as both **WebElement** and **WebDriver** implement the **SearchContext** interface.

- The **SearchContext** interface defines two abstract methods, **findElement(By by)** and **findElements(By by)**, the former returns a single **WebElement** whilst the latter returns a **List<WebElement>**.

```
public interface SearchContext {
    WebElement findElement(By by);
    List<WebElement> findElements(By by);
}
```

Element Selection Strategies (By API)

Static method	Returned type (All are returned in a By object variable)	Description
By.id(String id)	**ById**	Creates a **ById** selector for selecting an element by its ID.
By.linkText(String linkText)	**ByLinkText**	Creates a **ByLinkText** selector for selecting anchor elements by the exact text they display. If more than one is present, selects the first match.
By.partialLinkText(String partialLinkText)	**ByPartialLinkText**	Creates a **ByPartialLinkText** selector for selecting anchor elements that contain the given text.
By.name(String name)	**ByName**	Creates a **ByName** selector for selecting elements with a matching **name** attribute.

Element Selection Strategies (By API) cont...

Static method	Returned type (All are returned in a By object variable)	Description
By.tagName(String tagName)	**ByTagName**	Creates a **ByTagName** selector that selects elements with the matching tag name.
By.xpath(String xpathExpression)	**ByXPath**	Creates a **ByXPath** selector that selects elements via xpath expressions.
By.className(String className)	**ByClassName**	Creates a **ByClassName** selector that selects elements with a matching class attribute value.
By.cssSelector(String cssSelector)	**ByCssSelector**	Creates a **ByCssSelector** that selects elements via a CSS expression. Not every browser may implement the Selector API defined by the W3C, Selenium aims for at least CSS2 support with this selector type.

Element Selection Strategies (Examples)

Java

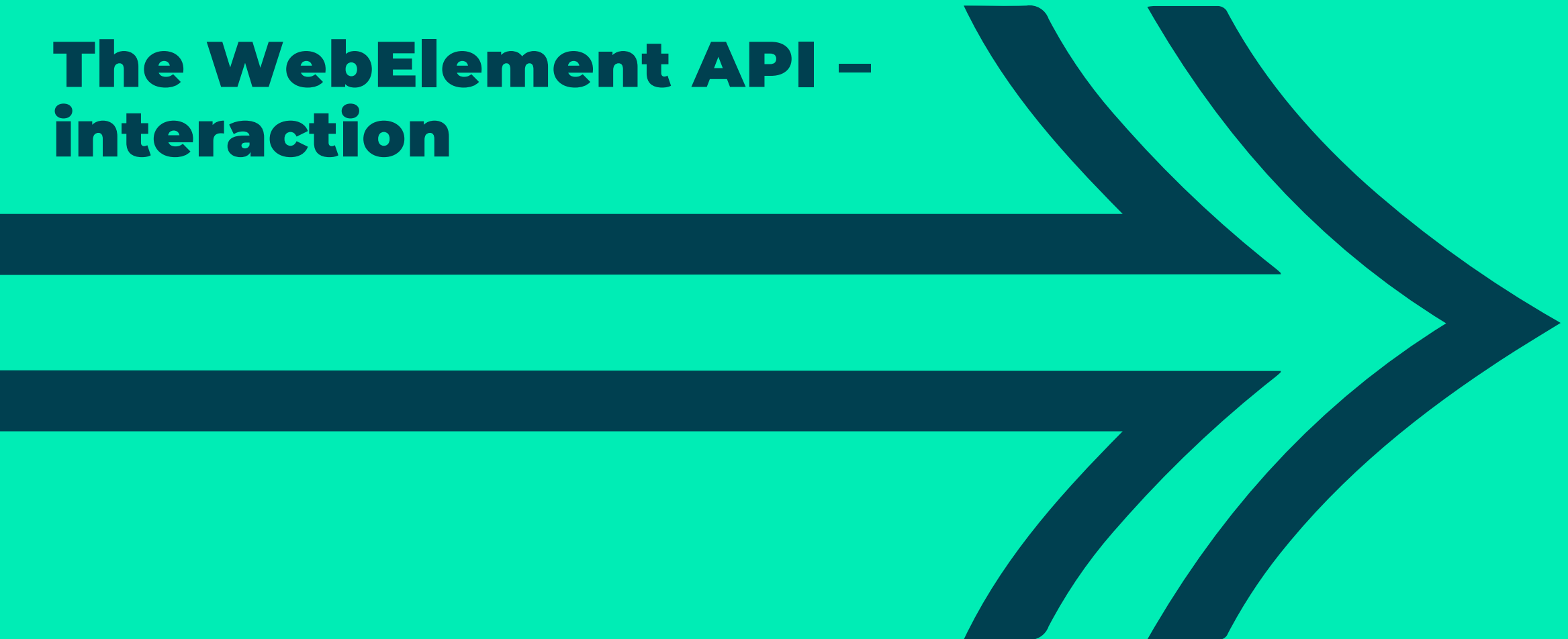
driver.findElement(...)

By.id	.findElement(By.id("profile-picture")	
By.cssSelector	.findElement(By.cssSelector("body > div:nth-child(2)"))	<body> • • <--
By.xpath	.findElement(By.xpath("/html/body/div[2]"))	<body> • • <--
By.name	.findElement(By.name("searchbar"))	<input name="searchbar"/>
By.className	.findElement(By.className("btn"))	<input class="btn btn-primary">
By.tagName	.findElement(By.tagName("header"))	<header> Welcome to QA </header>
By.linkText	.findElement(By.linkText("About"))	About
By.partialLinkText	.findElement(By.partialLinkText("out"))	About

Exercise 5 & 6

See exercises 5 and 6 in your Selenium WebDriver exercise book.

The WebElement API – interaction



WebElement Instance Methods (API)

Once we have a WebElement, we are given a nice API to work with that allows us to interact with and retrieve information about a web element or its descendants. Methods available on the API include:

Methods	Return type	Description
WebElement.findElement(By by)	**WebElement**	Finds sub element in selected element. For instance, if the selected element was a list, might want to find all list items in that list. Affected by implicit wait times.
WebElement.findElements(By by)	**List<WebElement>**	Returns a list of web elements using the supplied selector. Affected by implicit wait times, will return as soon as the List contains more than 0 elements.
WebElement.click()	**void**	Clicks the element. If clicking an element loads a new page, discards the object reference in use.
WebElement.getText()	**String**	Gets the visible text of an element and its children (sub-elements), does not include text hidden by CSS.

WebElement Instance Methods (API) cont...

Methods	Return type	Description
WebElement.getAttribute(String name)	**String**	Returns the value of the specified attribute.
WebElement.sendKeys(CharSequence... keys)	**void**	Simulates typing into an element.
WebElement.submit()	**void**	Submits a form to a server, has blocking behaviour if a page load is triggered.
WebElement.clear()	**void**	If the element is an **input**, empties the text content in its **value** property.

WebElement API (Example)

The following example shows how we might send input to an input field:

```
@Test
public void interactingWithElements() {
   driver.get("https://www.bing.com");
   WebElement searchBar = driver.findElement(By.name("q"));
   searchBar.sendKeys(Keys.chord(Keys.SHIFT, "puppies"));
   searchBar.sendKeys(" and kittens");
   searchBar.submit();
}
```

- The **sendKeys()** method accepts a **CharSequence** as input allowing us to directly supply a string.
- We can also use the **Keys.chord()** static method to simulate a user action like holding the shift key while typing to make some inputs case invert.
- We use the **submit()** method to submit the form that the search bar is inside.

Exercise 7

See exercise 7 in your Selenium WebDriver exercise booklet.


Thank you!
Hope you enjoyed this learning journey.