SOFTWARE TESTING
Selenium WebDriver 1

HTML

- **HTML** stands for Hyper Text Markup Language
- **HTML** describes the structure of Web pages
- **HTML** elements are the building blocks of HTML pages
- **HTML** elements are represented by tags
- **HTML** tags label include the following: "heading", "paragraph", "table“ etc
- is the most widely used language on Web to develop web pages.

HTML Tags

- **<!DOCTYPE html>** stands for Hyper Text Markup Language
- **<html>** This tag encloses the complete HTML document
- **<head>** This tag represents the document's header which can keep other HTML tags like <title>
- **<title>** is used inside the <head> tag to mention the document title.
- **<body>** represents the document's body which keeps other HTML tags like <h1>, , <p> etc.
- **<h1>** his tag represents the heading
- **<p>** represents a paragraph

HTML Sample Page

```
<!DOCTYPE html>
<html>
  <head>
    <title>ITSTEP Example</title>
  </head>

  <body>
    <h1>This is heading 1</h1>
      <p> This is a paragraph which represents just basic text on
      a web page</p>
    <h2>This is heading 2</h2>
      <p> This is another paragraph which represents just basic text on
      a web page</p>
    <h3>This is heading 3</h3>
    <p> This is the last paragraph which represents just basic text on
      a web page</p>
  </body>

</html>
```

HTML Tags

- **
** Line Break Tag, anything following it starts from the next line, you do not need opening and closing tags.

```
<!DOCTYPE html>
<html>

  <head>
    <title>Line Break  Example</title>
  </head>

  <body>
    <p>Hello<br />
      You delivered your assignment on time.<br />
      Thanks<br /> Group</p>
  </body>

</html>
```

HTML Tags

- ** ** put any content in the center of the page

```
<!DOCTYPE html>
<html>

  <head>
    <title>Centring Content Example</title>
  </head>

  <body>
    <p>This text is not in the center.</p>

    <center>
      <p>This text is in the center.</p>
    </center>
  </body>

</html>
```

HTML Tags

- **<hr>** tag creates a line from the current position in the document to the right

  For example, you may want to give a line between two paragraphs as in the given example below:

```
<!DOCTYPE html>
<html>

  <head>
    <title>Horizontal Line Example</title>
  </head>

  <body>
    <p>This is paragraph one and should be on top</p>
    <hr />
    <p>This is paragraph two and should be at bottom</p>
  </body>

</html>
```

HTML Tags

- **<hr /> (horizontal lines)** tag creates a line from the current position in the document to the right

  For example, you may want to give a line between two paragraphs as in the given example below:

```
<!DOCTYPE html>
<html>

  <head>
    <title>Horizontal Line Example</title>
  </head>

  <body>
    <p>This is paragraph one and should be on top</p>
    <hr />
    <p>This is paragraph two and should be at bottom</p>
  </body>

</html>
```

HTML Tags

- **<i>** tag stands for *Italic* and is usually applied to text in paragraphs
- **<u>** tag stands for underline and is also applied to text
- **<b>** tag stands for **bold**

```
<!DOCTYPE html>
<html>
  <head>
    <title>ITSTEP Example</title>
  </head>
  <body>
    <h1>This is heading 1</h1>
      <p><i> This is a paragraph which represents just basic text on
      a web page</i></p>
  <h2>This is heading 2</h2>
  <p> <b>This is another paragraph which represents just basic text on
      a web page</b></p>
    <h3>This is heading 3</h3>
    <p> <u>This is the last paragraph which represents just basic text on
      a web page</u></p>
  </body>

</html>
```

Tag Attributes

- **An attribute is used to define the characteristics of an HTML element and is placed inside the element's opening tag.**

```
<!DOCTYPE html>
<html>

  <head>
    <title>Align Attribute  Example</title>
  </head>

  <body>
    <p align = "left">This is left aligned</p>
    <p align = "center">This is center aligned</p>
    <p align = "right">This is right aligned</p>
  </body>

</html>
```

Tag Attributes

- The four core attributes that can be used on the majority of HTML elements:
  - **id** - attribute of an HTML tag can be used to uniquely identify any element within an HTML page

    ```
    <p id = "html">This para explains what is HTML</p>
    <p id = "css">This para explains what is Cascading Style Sheet</p>
    ```

  - **title** - attribute gives a suggested title for the element

    ```
    <h3 title = "Hello HTML!">Titled Heading Tag Example</h3>
    ```

  - **class** - attribute is used to associate an element with a style sheet, and specifies the class of element

    ```
    <div class="w3-container top"> ..... </div>
    ```

  - **style** - allows you to specify Cascading Style Sheet (CSS) rules within the element

    ```
    <p style = "font-family:arial; color:#FF0000;">Some text...</p>
    ```

Images

- **<img>** - You can insert any image in your web page by using this tag

  The **<img>** tag is an empty tag, which means that, it can contain only list of attributes and it has no closing tag.

```
<!DOCTYPE html>
<html>

 <head>
    <title>Using Image in Webpage</title>
 </head>

<body>
    <p>Simple Image Insert</p>
    <img src = "/html/images/test.png" alt = "Test Image" />
</body>

</html>
```

Tables

- The HTML tables are created using the <table> tag in which the <tr> tag is used to create table rows and <td> tag is used to create data cells.

```
<body>
    <table border = "1">
      <tr>
        <td>Row 1, Column 1</td>
        <td>Row 1, Column 2</td>
      </tr>

      <tr>
        <td>Row 2, Column 1</td>
        <td>Row 2, Column 2</td>
      </tr>
    </table>
 </body>
```

- Here, the **border** is an attribute of <table> tag and it is used to put a border across all the cells.

Lists

- **<ul>** - An unordered list. This will list items using plain bullets.

- **<li>** - an item in the list

```
<body>
    <ul type = “square”>
      <li>Beetroot</li>
      <li>Ginger</li>
      <li>Potato</li>
      <li>Radish</li>
    </ul>
  </body>
```

- You can use **type** attribute for <ul> tag to specify the type of bullet you like. By default, it is a disc. Following are the possible options:

```
<ul type = "square">
<ul type = "disc">
<ul type = "circle">
```

<ol> - An ordered list.

Link

- A link is specified using HTML tag <a>. This tag is called **anchor tag** and anything between the opening <a> tag and the closing tag becomes part of the link.

```
<!DOCTYPE html>
<html>

   <head>
    <title>Hyperlink Example</title>
   </head>

   <body>
      <p>Click following link</p>
      <a href = "https://www.google.com">Google</a>
   </body>

</html>
```

Forms

- HTML Forms are required, when you want to collect some data from the site visitor.

```
<!DOCTYPE html>
<html>

  <head>
    <title>Text Input Control</title>
  </head>

  <body>
    <form >
      First name: <input type = "text" name = "first_name" />
      <br>
      Last name: <input type = "text" name = "last_name" />
    </form>
  </body>

</html>
```

CSS Style

- Cascading Style Sheets (CSS) describe how documents are presented on screens

```
<!DOCTYPE html>
<html>

  <head>
    <title>HTML CSS</title>
  </head>

  <body>
    <p style = "color:green; font-size:24px;" >Hello, World!</p>
  </body>

</html>
```

CSS Style

- Cascading Style Sheets (CSS) describe how documents are presented on screens

```
<!DOCTYPE html>
<html>

  <head>
    <title>HTML CSS</title>
  </head>

  <body>
    <p style = "color:green; font-size:24px;" >Hello, World!</p>
  </body>

</html>
```

CSS Style

```
<!DOCTYPE html>
<html>

<body style="background-color:powderblue;">

<h1>This is a heading</h1>
<p>This is a paragraph.</p>

</body>
</html>
```

CSS Style

```
<!DOCTYPE html>
<html>

<body style="background-color:powderblue;">

<h1>This is a heading</h1>
<p>This is a paragraph.</p>

</body>
</html>
```

XPath

- **XPath** is a query language used commonly to search particular elements or attributes with matching patterns.
- It defines a language to find information in an XML file.
- XPath uses a path expression to select node or a list of nodes from an XML document.

XPath

1	**node-name** Select all nodes with the given name "nodename"
2	**/** Selection starts from the root node
3	**//** Selection starts from the current node that match the selection
4	**.** Selects the current node
5	**..** Selects the parent of the current node
6	**@** Selects attributes

XPath Syntax

```
<h1>This is heading 1</h1>
      <button id = "btn" border = "2"> Button </button>

      <h2>This is heading 2
        <button border = "2"> FirstButton </button>
        <button border = "2"> SecondButton </button>
      </h2>
```

Select an element by id:

//button[@id='btn'] - select the Button

//h2//button – returns 2 buttons from the header h2

//h2//button[text()='FirstButton'] – selects the first button from h2

//h2//button[contains(text(), 'Second')] – selects the second button from h2

XPath Syntax

```
<ul id = "firstList">
      <li>Text1</li>
      <li>Text2</li>
      <li>Text3</li>
      <li>Text4</li>
     </ul>

<ul id = "secondList">
      <li>Text1</li>
      <li>Text2</li>
      <li>Text3</li>
      <li>Text4</li>
      </ul>
```

//li[text() = ‘Text1’] - selects 2 options from both lists

//ul[@id = ‘firstList’]//li[text() = ‘Text1’] - selects option with Text1 from 1 list

//ul[@id = ‘secondList’]//li[text() = ‘Text1’] - selects option with Text1 from 2 list

Selenium WebDriver

A test tool that allows you to write automated web application UI tests in any programming language against any HTTP website using any mainstream JavaScript-enabled browser

```
System.setProperty("webdriver.gecko.driver","path of geckodriver.exe");
WebDriver driver = new FirefoxDriver();
```

```
System.setProperty("webdriver.chrome.driver", "path of chromedriver.exe");
WebDriver driver = new ChromeDriver();
```

Selenium WebDriver

```
WebDriver driver = new FirefoxDriver();  // a driver object is instantiated

driver.get("https://www.google.com/");
```

get() method is used to launch a new browser session and directs it to the URL that you specify as its parameter

```
driver.close();
```

close() method is used to close the browser instance.

Finding WebElements

By id:

```
<div id="coolestWidgetEvah">...</div>
```

java

```
WebElement element = driver.findElement(By.id("coolestWidgetEvah"));
```

By class name:

```
<div class="cheese"><span>Cheddar</span></div><div class="cheese"><span>Gouda</span></div>
```

java

```
List<WebElement> cheeses = driver.findElements(By.className("cheese"));
```

Finding WebElements

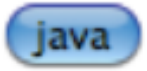
By link text:

```
<a href="http://www.google.com/search?q=cheese">cheese</a>>
```


```
WebElement cheese = driver.findElement(By.linkText("cheese"));
```

By partial link text:

```
<a href="http://www.google.com/search?q=cheese">search for cheese</a>>
```




```
WebElement cheese = driver.findElement(By.partialLinkText("cheese"));
```

Finding WebElements

By tag name:

```
<iframe src="..."></iframe>
```

java

```
WebElement frame = driver.findElement(By.tagName("iframe"));
```

By name:

```
<input name="cheese" type="text"/>
```

java

```
WebElement cheese = driver.findElement(By.name("cheese"));
```

Finding WebElements

By XPath:

```
<input type="text" name="example" />
<INPUT type="text" name="other" />
```


```
List<WebElement> inputs = driver.findElements(By.xpath("//input"));
```

Managing the browser instance

```
WebDriver driver = new FirefoxDriver();

WebElement myButton = driver.findElement(By.xpath(“//h2//button[text()=‘FirstButton’]”));

driver.get(www.gotosomesite.com);

myButton.click();

driver.manage().window().maximize(); // maximizes the browser window size

driver.navigate().to("http://www.example.com"); //navigates to the URL specified

driver.navigate().forward(); //navigates forward

driver.navigate().back(); // navigates back

driver.close() // closes the browser window
```

Driver Actions

```
WebDriver driver = new FirefoxDriver();

WebElement myButton = driver.findElement(By.xpath(“//h2//button[text()=‘FirstButton’]”));

WebElement myField = driver.findElement(By.xpath(“//h2//input[text()=‘Name’]”));

driver.get(www.gotosomesite.com);

myButton.click(); // method clicks in the middle of element

myField.sendKeys(“John”); // types into the WebElement (field)

myField.getText(); //returns the text from the WebElement
```

Driver Actions

`Action`	`build()` Generates a composite action containing all actions so far, ready to be performed (and resets the internal builder state, so subsequent calls to `build()` will contain fresh sequences).
`Actions`	`click()` Clicks at the current mouse location.
`Actions`	`click(WebElement target)` Clicks in the middle of the given element.
`Actions`	`clickAndHold()` Clicks (without releasing) at the current mouse location.
`Actions`	`clickAndHold(WebElement target)` Clicks (without releasing) in the middle of the given element.
`Actions`	`contextClick()` Performs a context-click at the current mouse location.
`Actions`	`contextClick(WebElement target)` Performs a context-click at middle of the given element.
`Actions`	`doubleClick()` Performs a double-click at the current mouse location.
`Actions`	`doubleClick(WebElement target)` Performs a double-click at middle of the given element.
`Actions`	`dragAndDrop(WebElement source, WebElement target)` A convenience method that performs click-and-hold at the location of the source element, moves to the location of the target element, then releases the mouse.

Driver Actions

Actions	**moveByOffset**(int xOffset, int yOffset) Moves the mouse from its current position (or 0,0) by the given offset.
Actions	**moveToElement**(**WebElement** target) Moves the mouse to the middle of the element.
Actions	**moveToElement**(**WebElement** target, int xOffset, int yOffset) Moves the mouse to an offset from the top-left corner of the element.
Actions	**pause**(java.time.Duration duration)
Actions	**pause**(long pause) Performs a pause.
void	**perform**() A convenience method for performing the actions without calling build() first.
Actions	**release**() Releases the depressed left mouse button at the current mouse location.
Actions	**release**(**WebElement** target) Releases the depressed left mouse button, in the middle of the given element.
Actions	**sendKeys**(java.lang.CharSequence... keys) Sends keys to the active element.
Actions	**sendKeys**(**WebElement** target, java.lang.CharSequence... keys) Equivalent to calling: *Actions.click(element).sendKeys(keysToSend)*. This method is different from **WebElement.sendKeys(CharSequence...)** - see **sendKeys(CharSequence...)** for details how.

JUnit

- JUnit is an open source framework, which is used for writing and running tests.
- Provides annotations to identify test methods.
- Provides assertions for testing expected results.
- JUnit shows test progress in a bar that is green if the test is running smoothly, and it turns red when a test fails.

JUnit (Unit Test Example)

- In order to define a JUnit test add a @Test annotation to the java method that will be representing the test.
  Example:

```
public int calculateSum(int a, int b){
      return a+b;
}

@Test
public void testCheckSum(){
        assertEquals(6, calculateSum(2,3));
 }
```

JUnit (WebDriver Test)

- So, what can we check ?
  - Current URL - driver.getCurrentUrl();
  - Text on page - element.getText();
  - Error messages - element.getText();
  - Page title - driver.getTitle();
  - Value of fields - element.getAttribute("value");

JUnit (WebDriver Test)

- Every test should compare actual results with expected results

- We do this by using assertEquals method of JUnit framework
  Example:

  - assertEquals(expectedResult, actualResult);
    // returns false if the values are not equal and true if the values are equal