EU Fast Track Selenium Day 3

▼ Waits

- Sometimes we will get NoSuchElementException, this may cause from several reasons:
  - locator is wrong
  - Synchronization: when the driver and browser are not at the same page

- Solutions for Synchronization problems:
  1. Sleep
     a. Thread.sleep(milliseconds) —> will stop the whole code for given time, not recomended
     b. Must extends from Thread class
     c. Comes from Java

```
class TestSleepMethod1 extends Thread{
 public void run(){
  for(int i=1;i<5;i++){
  // the thread will sleep for the 500 milli seconds
    try{Thread.sleep(500);}
    catch(InterruptedException e){System.out.println(e);}
    System.out.println(i);
  }
}
```

  2. Implicit Wait
     a. If the web element doesn’t appear, it will wait specific time
     b. If the element appears, it will not wait until end
     c. Comes from Selenium

```
driver.manage().timeouts().implicitlyWait(Time, TimeUnit.SECONDS);
```

3. Explicit Wait
    a. Wait for specific condition until maximum waiting time
    b. We have to create Object first
    c. Comes from Selenium

```
WebDriverWait wait=new WebDriverWait(driver, Time);
wait.until(ExpectedConditions.visibilityOfElementLocated(Web element));
elementToBeClickable;
elementToBeSelected;
titleContains;
...
...
...
```

4. Fluent Wait
    a. Try to find the element every specific time under maximum time
    b. We have to create Object first

```
Wait<WebDriver> wait = new FluentWait<WebDriver>(driver)
  .withTimeout(30, TimeUnit.SECONDS)
  .pollingEvery(5, TimeUnit.SECONDS)
  .ignoring(NoSuchElementException.class);
```

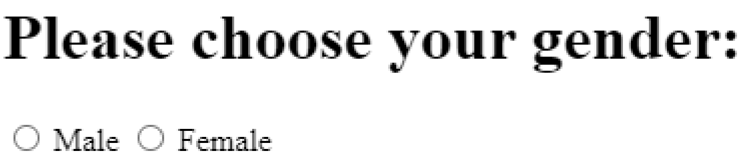
▼ Inputs

- Radio Buttons : allows users to select only one option

```
<!DOCTYPE html>
<html>
  <head>
    <title>Hello, World!</title>
    <link rel="stylesheet" href="styles.css" />
  </head>
  <body>
    <h1> Please choose your gender:</h1>
```

```
      <input type="radio" value="Male" /> Male
      <input type="radio" value="Female" /> Female

  </body>
</html>
```

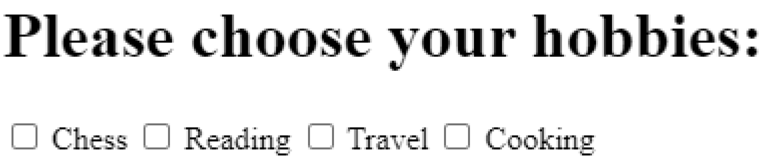


Please choose your gender:

◯ Male ◯ Female

- Check box : allows users to select multiple options

```
<!DOCTYPE html>
<html>
  <head>
    <title>Hello, World!</title>
    <link rel="stylesheet" href="styles.css" />
  </head>
  <body>
    <h1> Please choose your hobbies:</h1>
      <input type="checkbox" value="Chess" /> Chess
      <input type="checkbox" value="Reading" /> Reading
      <input type="checkbox" value="Travel" /> Travel
      <input type="checkbox" value="Cooking" /> Cooking
  </body>
</html>
```



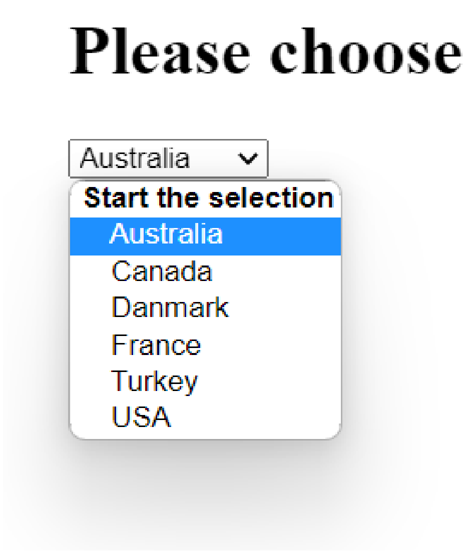
Please choose your hobbies:

☐ Chess ☐ Reading ☐ Travel ☐ Cooking

▼ Dropdown

- We have two types of Dropdown, html and Select type.
- If it’s not used Select tag, we call it html dropdown. (maybe used <li> tag)
- If it’s used Select tag, we call it Select dropdown

```
<!DOCTYPE html>
<html>
  <head>
    <title>Hello, World!</title>
    <link rel="stylesheet" href="styles.css" />
  </head>
  <body>
    <h1>Please choose your country</h1>
    <div class="container">
<select id="search-pax" name="pax" class="ls-select ">
<optgroup label="Start the selection">
  <option value="1">Australia</option>
  <option value="2">Canada</option>
  <option value="3">Danmark</option>
  <option value="4">France</option>
  <option value="5">Turkey</option>
  <option value="6">USA</option>
  </optgroup>
  </select>
</div>
  </body>
</html>
```


Please choose your country

- When handling HTML dropdown, we just locate the element, then do actions on it.
- When handling Select dropdown, first we locate that dropdown, then create object from Select class, then pass the dropdown web element to select object.

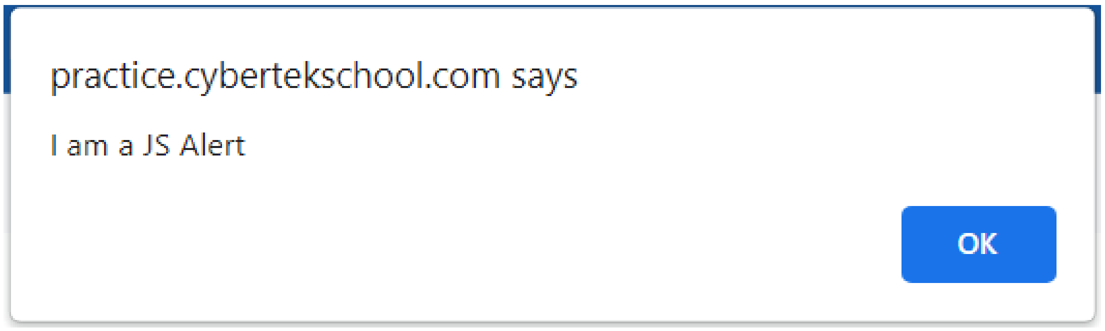
```
WebElement dropdown = driver.findElement(By.id("dropdown"));
Select select = new Select(dropdown);
select.selectByIndex(3);

// or we can directly pass the locator to select object
Select dropdownList = new Select(driver.findElement(By.id("country list"))
```

- Related methods:
    - getOptions() —> return all the options as a list
    - getFirstSelectedOption() —> to get first selected option
    - getAllSelectedOption() —> get all selected options
    - deselectAll() —> clear all selected entries
    - selectByVisibleText(String text) —> select by provided text
    - selectByValue(String value) —> select by html value tag
    - selectByIndex(int index) —> select by index number of options(index starts from 0)

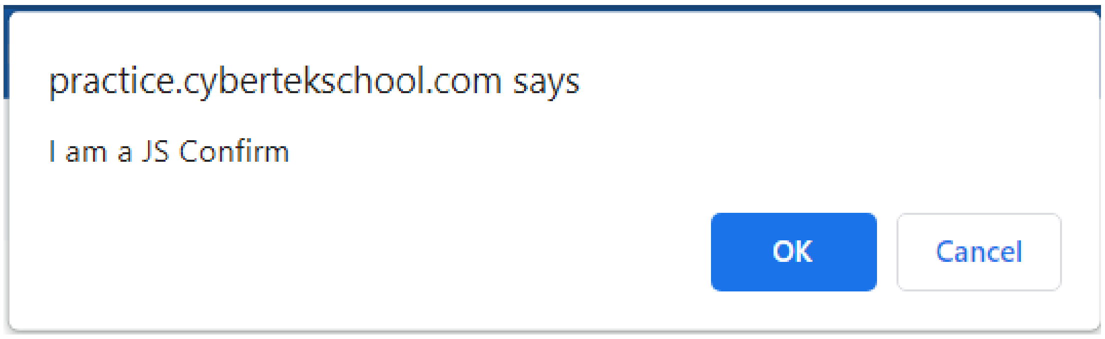
▼ Alerts

- HTML Alert
    - Comes from html
    - We can locate and handle them directly
- Java Script Alerts
    - Comes from JS
    - It will block the page, you have to deal with it before do anything else
    - 3 types of alert: Alert, Confirm and Prompt


practice.cybertekschool.com says

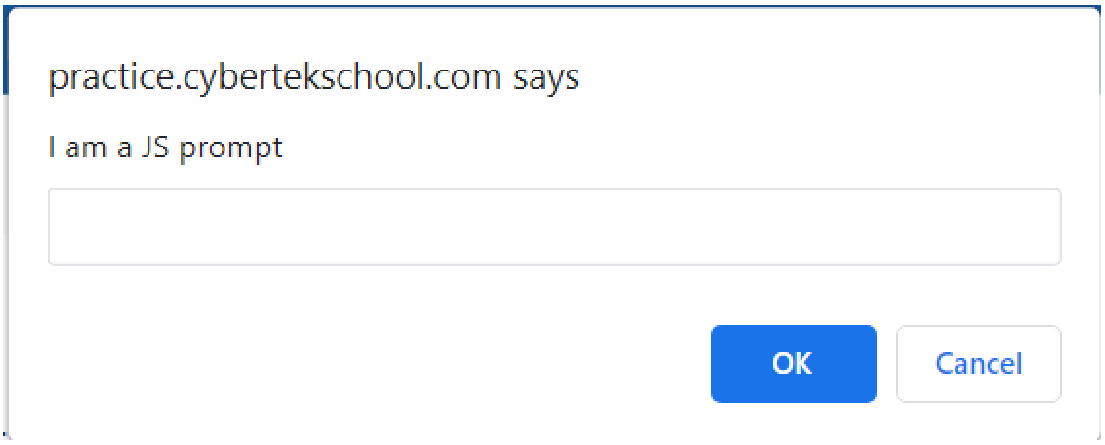
I am a JS Alert

Alert


practice.cybertekschool.com says

I am a JS Confirm

Confirm


practice.cybertekschool.com says

I am a JS prompt

Prompt

- How to handel?

```
http://practice.cybertekschool.com/javascript_alerts

Alert alert = driver.switchTo().alert();

alert.accep();
alert.dismiss():
alert.getText(); // capture alert message
alert.sendKeys("Text");
```

▼ Windows

- In Selenium, tab or window is same.
- Selenium can deal with one window at a time
- We have to switch to windows that we want to use through window handles
- Each window has a unique ID as a string

```
driver.getWindowHandle(); // get one window handle
driver.getWindowHandles(); // get all window handles as a set of string
```

▼ Web Tables

- Arrange the data in row and columns
- Starts with <table> tag
- <th> stands for table head
- <tr> stands for table row
- <td> stands for cell data

```
http://practice.cybertekschool.com/web-tables

//tbody//div[@class='content']//td[3]
```

▼ Junit

- Framework that helps us run our test
- We will use different annotations according to our needs
- We don't have to use main() method when using Junit

▼ Annotations

- @Test
    - Convert a method to a test
    - Run by alphabetical order by default
    - Tests are independent from each other, failed tests will not affect next one

- @Before
    - Runs before every @Test method
    - Used for make some settings before every test

- @After
    - Runs after every @Test method
    - Used for end the settings before every test

- @BeforeClass
    - Runs once before all the methods in same class

- @AfterClass
    - Runs once after all the methods in same class

- @Ignore
    - Ignore the specific test

Differences between TestNG and JUnit

https://www.softwaretestinghelp.com/junit-vs-testng/