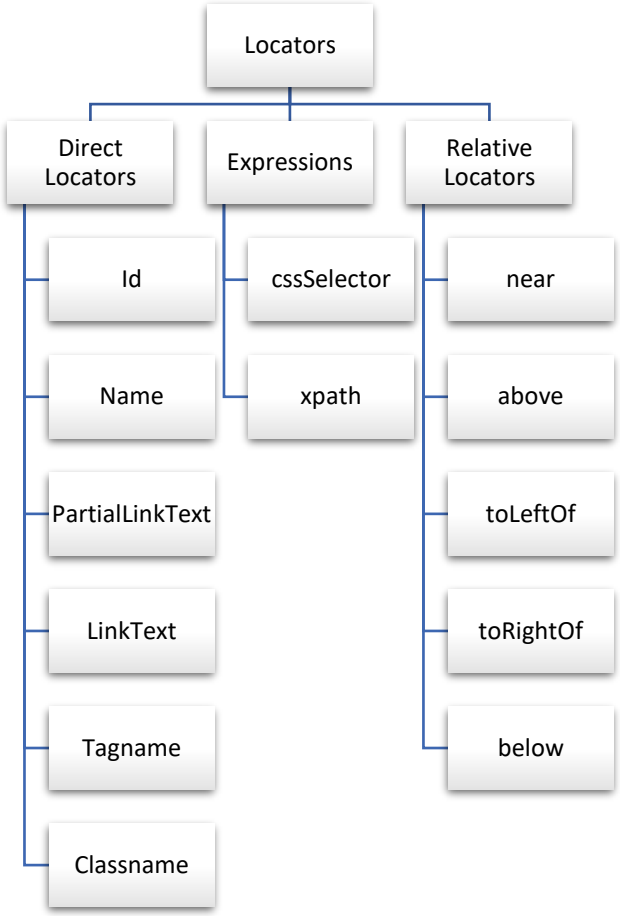
```
driver.findElement(By.id(" "));
driver.findElement(By.name(" "));
driver.findElement(By.linkText(" "));
driver.findElement(By.partialLinkText(" "));
driver.findElement(By.tagName(" "));
driver.findElement(By.className(" "));
driver.findElement(By.cssSelector(" "));
driver.findElement(By.xpath(" "));
```

Locators

It is used to identify/locate the web elements.

Locators are of mainly 3 types.



DIRECT LOCATORS

1. **id():**
   It is used to identify an element using the "id" attribute present in the html code.
   Syntax:

```
driver.findElement(By.id(" "));
```

   It is given the topmost priority.
   id() is the fastest locator because it unique.
   sendKeys():
   It is used to pass some data to the textfield.
   click():
   It is used to perform the click operation.

2. **name():**
   It is used to identify an element using the "name" attribute present in the HTML code.
   Syntax:

```
driver.findElement(By.name(" "));
```

   It is given the Second most priority.

3. **linkText():**
   It is used to identify an element using the text which is present between anchor tag.
   <a>Text--->Link text
   Text --->visible text/ Text of the web element
   It is given 3rd priority.
   Syntax:

```
driver.findElement(By.linkText(" "));
```

4. **partialLinkText():**
   It is used to identify/ locate the web elements by using the part of the text present between the anchor tag.

   When do we go for partial link text-
   1. Link is very lengthy

2. link is partially dynamic.
Ex: <a>Selenium 2025
3. Whenever we have blank spaces at beginning or at the end of the text.
Syntax:

```
driver.findElement(By.partialLinkText(" "));
```

It is given the 3rd priority.

5. **tagName():**
It is used to identify an element by using the tag name.
It is not recommended, because of multiple elements will be developed using same tag name.
when to use tag name locator:
-to identify the total no of elements present in a webpage.
Ex: Count the total no of links in a webpage
Count the total no of images in a webpage.
Since, we are fetching multiple elements, we have to use findElements().
Syntax:

```
driver.findElements(By.tagName(" "));
```

6. **className():**
It is used to identify an element by using the class attribute present in the html code.
Syntax:

```
driver.findElement(By.className(" "));
```

It is least recommended, because-
1. It is dynamic in nature (when UI changes, class name will get changed).
2. It is alphanumeric.
3. It will be compound (abc edf-rfg ihg).
Whenever we use the compound class name as a locator, compiler will throw "InvalidSelectorException".

EXPRESSIONS

1. **cssSelector():**
   It is used to identify an element using the following
   syntax:

```
tagName[AttributeName='AttributeValue']
```

   Tag must contain atleast 1 attribute.
   It is unidirectional.
   Ex:
   <input type="text" value="name" />
   driver.findElement(By.cssSelector("input[type='text']"));

2. **xpath():**
   It is one of the relative paths.
   It is multi directional.
   It is used to identify an element by using the following syntax:
   1. xpath by attribute:
   Syntax:

```
//tagName[@AttributeName='AttributeValue']
```

   2. xpath by text function/ xpath by visible text
   Synatx:

```
//tagName[text()='Text Value']
```

   3. xpath by contains
   Syntax:

```
//tagName[contains(@AN,'AV')]
Or
//tagName[contains(text(),'TV')]
```

   When to go for contains method-
   1. Whenever we have lengthy AV or TV.
   2. Whenever we have space at the beginning or the end of the AV or TV.
   3. Whenever we have dynamic AV or TV.
   4. Xpath by starts-with methods
   Syntax:

```
//tagName[starts-with(@AN,'AV')]
Or
//tagName[starts-with(text(),'TV')]
```

5. xpath by ends-with

Syntax:

```
//tagName[ends-with(@AN,'AV')]
Or
//tagName[ends-with(text(),'TV')]
```

6. xpath by multiple attributes

By making use of more than 1 attribute. We can locate the webelement, with
the help of logical operator[and, or, not].
-Here we are using and.
Here there are 3 ways-

1. *attribute and text()*

Syntax:

```
//tagName[@AN='AV' and text()='TV']
```

2. *text() and Attribute*

Syntax:

```
 //tagName[text()='TV' and @AN='AV']
```

3. *Attribute and attribute*

Syntax:

```
//tagName[@AN='AV' and @AN='AV']
```

7. xpath By surroundings

It is also known as dependent and independent xpath.
-Using surrounding element, the desired element is identified, it is known as xpath by surroundings.
Surroundings-->dependent element
-->independent element

Dependent element--> Anything which is duplicated will be considered as dependent.
Independent element---> Anything which has unique value they are independent.

To locate dependent and independent elements, we need to follow 4 steps:
Step1: Identify which is dependent and which is independent element.
Step2: Write xpath for independent element.
Step3: Traverse back to meet immediate common parent.
how to traverse back:
using "/.."
Whenever both independent and dependent get highlighted together, we should decide
that we have discovered the common parent.
Step4: update the xpath by writing the xpath for dependent element.

8. Xpath by axes

Syntax:

```
//tagName[@AN='AV']/axisname::tagName
or
//tagName[text()='TV']/axisname::tagName
```

::-->scope resolution operator
axis name-->1. following-sibling
2. preceding-sibling
3. ancestor

9. xpath By indexing:

It is not recommended,
It should be the last option when any of the xpath or locators are not working, we can
make use of this.
Syntax:

```
(xpath)[index]
```

Index starts from 1.

RELATIVE LOCATORS:

5 methods

- above()
- below()
- toLeftOf()
- toRightOf()
- near()

above()

Syntax:

WebElement ref=driver.findElement(By.locator);

driver.findElement(RelativeLocator.with(By.tagname(" ")).above(ref));

below()

Syntax:

WebElement ref=driver.findElement(By.locator);

driver.findElement(RelativeLocator.with(By.tagname(" ")).below(ref));

toLeftOf()

Syntax:

WebElement ref=driver.findElement(By.locator);

driver.findElement(RelativeLocator.with(By.tagname(" ")).toLeftOf(ref));

toRightOf()

Syntax:

```
WebElement ref=driver.findElement(By.locator);
driver.findElement(RelativeLocator.with(By.tagname(" ")).toRightOf(ref));
```

near()

Syntax:

```
WebElement ref=driver.findElement(By.locator);
driver.findElement(RelativeLocator.with(By.tagname(" ")).near(ref));
```