Bug Resistance

Automation Testing

Selenium

www.bugresistance.com

Selenium

Selenium a Web based automation testing tool that automates anything and everything available on a Web page. Initially started by Thought works & currently Google developers are supporting the latest version i.e WebDriver.

Selenium History

The Name Selenium came from a joke that Jason cracked once to his team. During Selenium's development, another automated testing framework was popular made by the company called Mercury Interactive (yes, the company who originally made QTP before it was acquired by HP). Since Selenium is a well-known antidote for Mercury poisoning, Jason suggested that name and his teammates took it. So that is how we got to call this framework up to the present.

Selenium Suit

Selenium is not just a single tool or a utility, rather a package of several testing tools and for the same reason, it is referred to as a Suite. Each of these tools is designed to cater to different testing and test environment requirements.

1

Selenium IDE

Selenium Integrated Development Environment (IDE) is the simplest framework in the Selenium suite and is the easiest one to learn. It is a Chrome and Firefox plugin that you can install as easily as you can with other plugins. However, because of its simplicity, Selenium IDE should only be used as a prototyping tool.

2

Selenium RC

Selenium RC was the flagship testing framework of the whole Selenium project for a long time. This is the first automated web testing tool that allows users to use a programming language they prefer. As of version 2.25.0, RC can support the following programming languages:
Java,C#,PHP,Python,Perl,Ruby

3

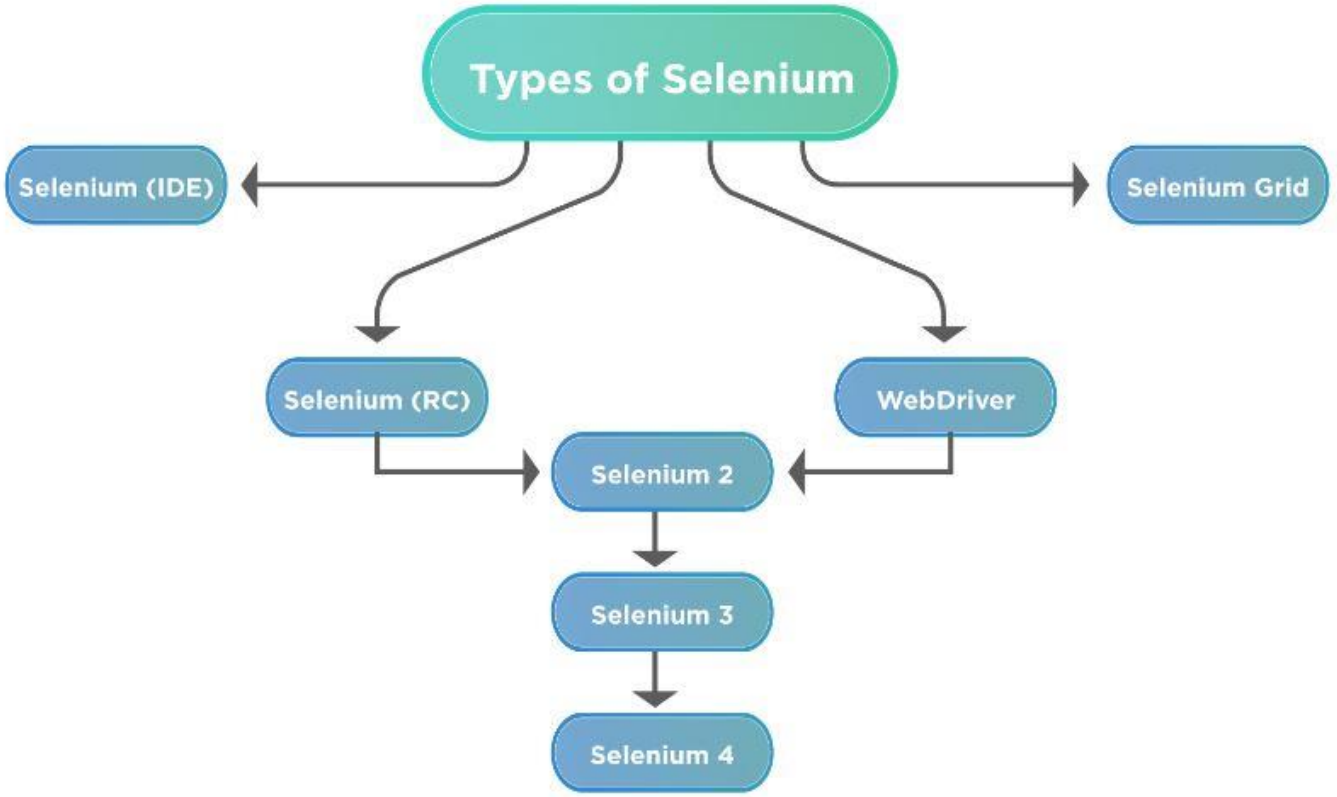
Selenium GRID

Selenium Grid is a tool used together with Selenium RC to run parallel tests across different machines and different browsers all at the same time. Parallel execution means running multiple tests at once.

4

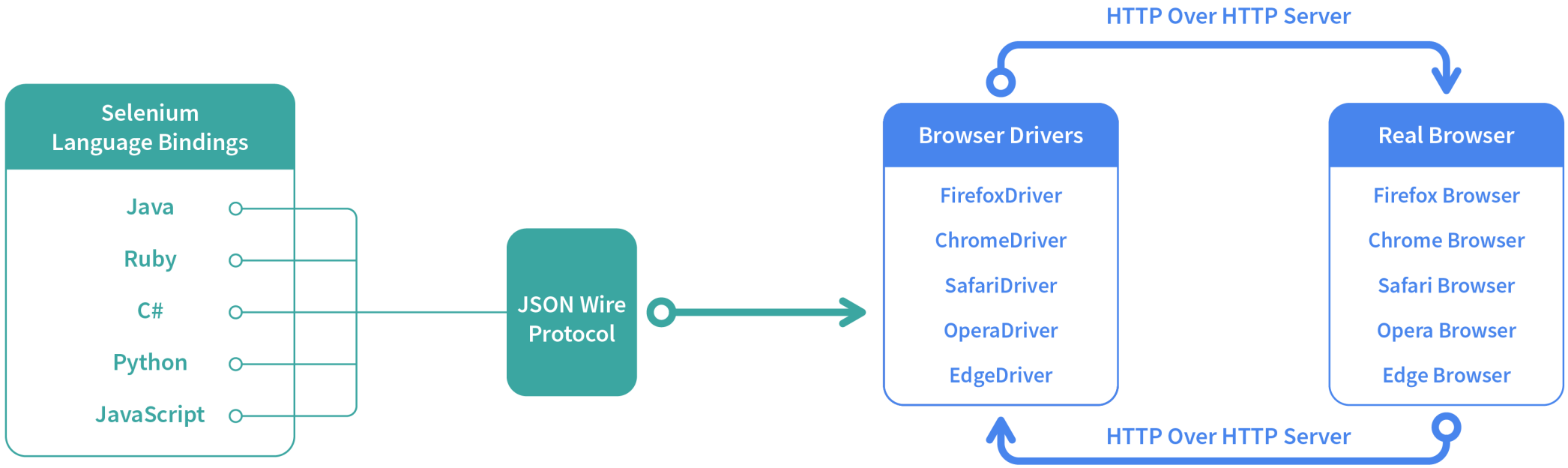
WebDriver

The WebDriver proves to be better than Selenium IDE and Selenium RC in many aspects. It implements a more modern and stable approach in automating the browser's actions. WebDriver, unlike Selenium RC, does not rely on JavaScript for Selenium Automation Testing. It controls the browser by directly communicating with it.



The whole Selenium Team decided to merge WebDriver and Selenium RC to form a more powerful tool called Selenium 2, with WebDriver being the core. Currently, Selenium RC is still being developed but only in maintenance mode. Most of the Selenium Project's efforts are now focused on Selenium 2.

Selenium Architecture



Selenium Limitation

Different Types Limitation in Selenium:

- Selenium does not support automation testing for desktop applications.
- Since Selenium is open source software, you have to rely on community forums to get your technical issues resolved.
- Captcha and Barcode readers cannot be tested using Selenium
- It is not possible to perform testing on images. We need to integrate Selenium with Sikuli for image based testing.
- Mobile applications cannot be tested using Selenium
- Selenium does not have any inbuilt reporting capability; you have to rely on plug-ins like JUnit and TestNG for test reports.
- We should know at least one of the supported programming languages to create test scripts in Selenium WebDriver.

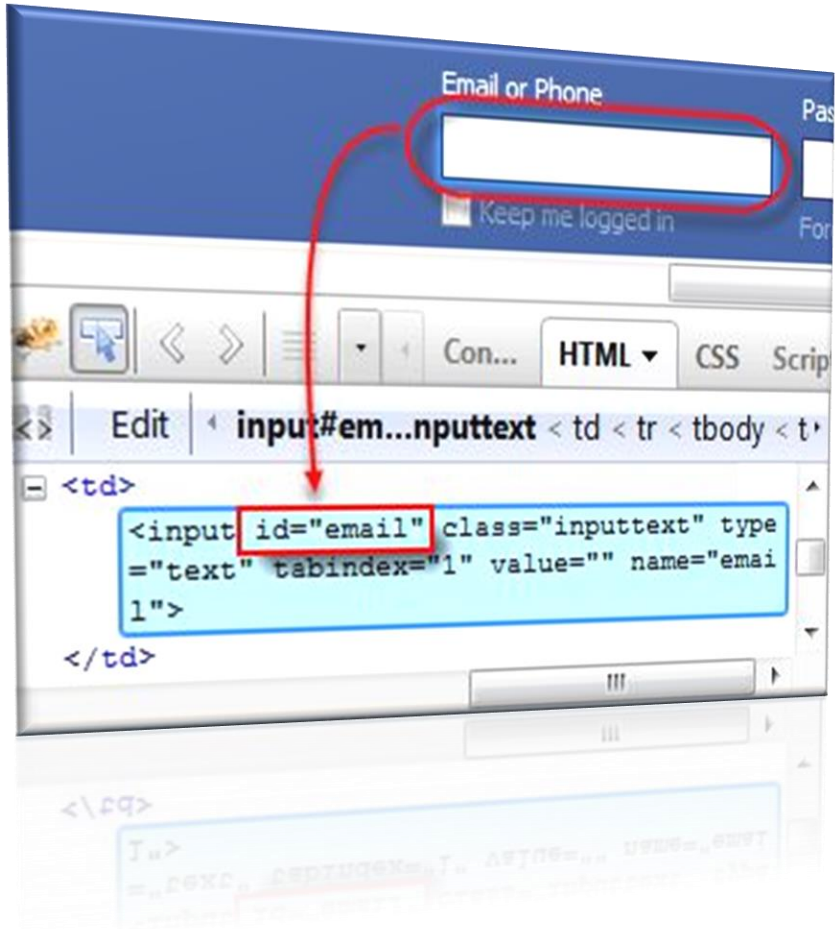
Locators

Locators are the way to identify an HTML element on a web page, and almost all UI automation tools provide the capability to use locators for the identification of HTML elements on a web page.

Different Types of Locators in Selenium:

- **By ID**: *find_element_by_id*
- **By Name**: *find_element_by_name*
- **By class name**: *find_element_by_class_name*
- **By name attribute**: *find_element_by_name*
- **By Xpath**: *find_element_by_xpath*
- **By CSS:** *find_element_by_css*

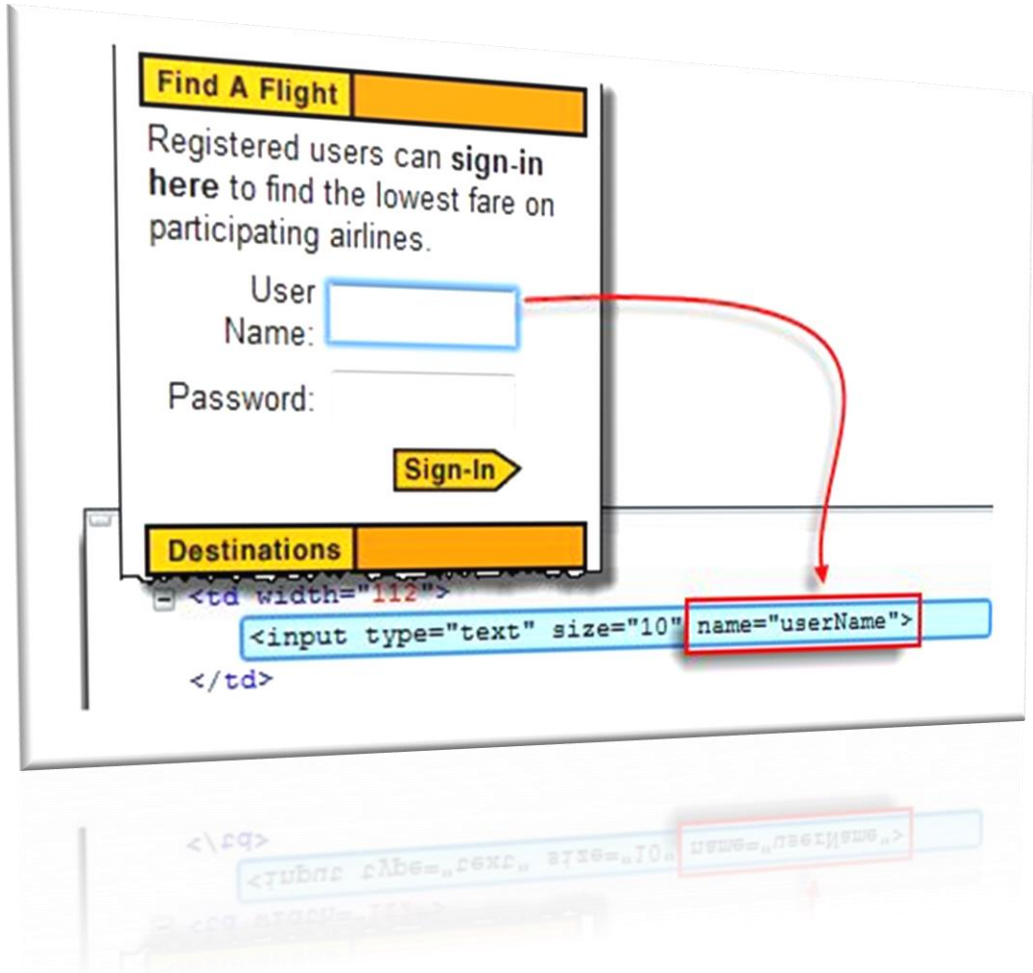
1.ID Locator



"***ID***" as a locator is one of the most common ways of identifying elements on a web page. According to ***W3C***, an ID for a web element always needs to be unique. The ***ID*** is one of the fastest and unique ways of locating web elements and is considered as one of the most reliable methods for determining an element. But with the advancement in technologies and more of the ***dynamic web pages***, "***IDs***" are generated dynamically and generally not the reliable way to locate a web element, as they change for different users.

- **driver. findElement(By. id("id value"))**

2.Name Locator



Selenium also offers users a way to identify the element using the ***name*** attribute. But contrary to id, a web page can have multiple elements with the same "***name***" attribute. So, if we are going to use the name attribute for the identification of a web element, we should always make sure that the name attribute must contain a unique value.

If multiple elements have the same value of the '*name*' attribute, then, Selenium will just select the first values in the page which matches the search criteria. So, we always recommend making sure that the value of the ***name*** attribute should be unique when we select it for locating a web element.

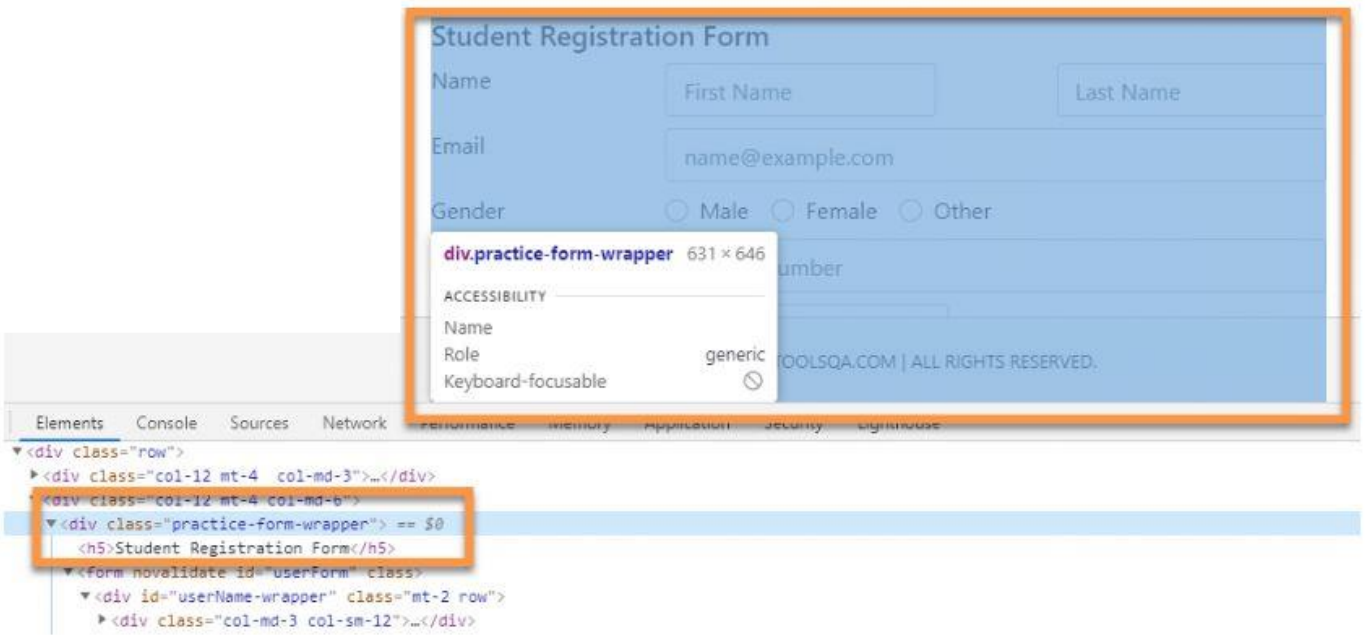
- **driver.findElement(By. name("name value"))**

3.Class Locator

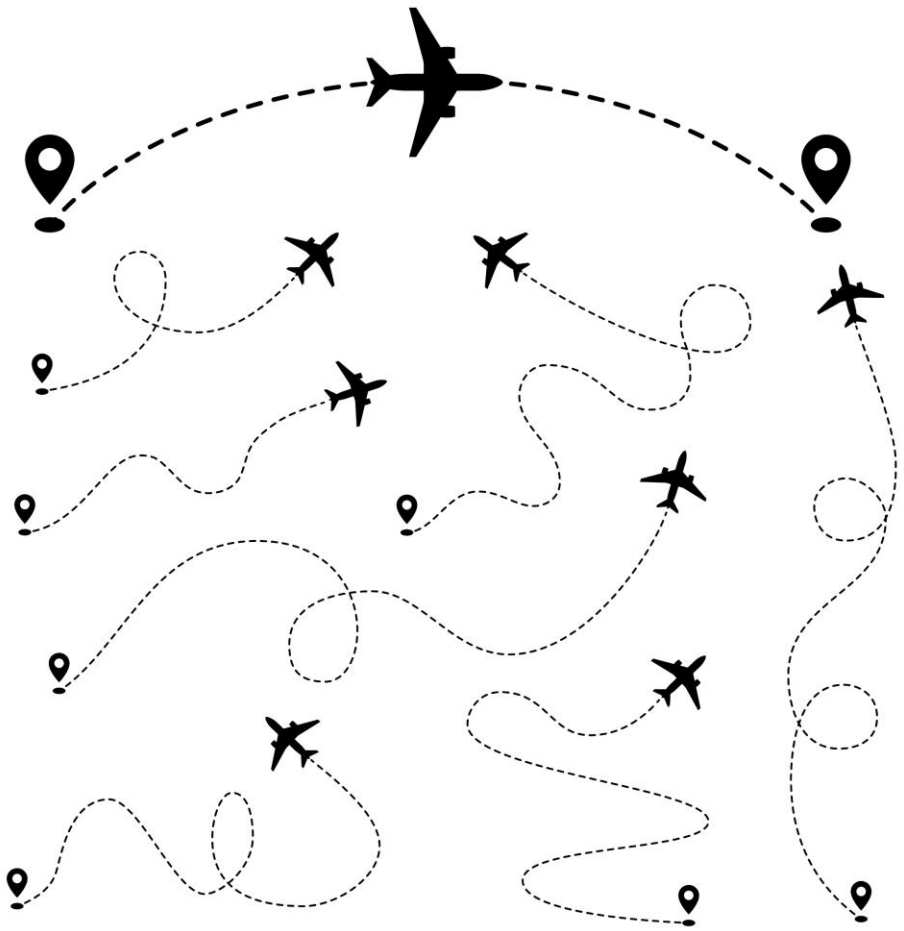
ClassName allows Selenium to find web elements based on the ***class*** values of the DOM. For example, if we want to identify or perform any operation on the form element, we can use the class to identify it.

To identify it successfully, we need to make sure that the ***class name*** value that we are using for locating the web element, i.e., web form, is unique, and any other class doesn't have the same value. If any other class contains the same value as this, then Selenium will select whichever element comes first while scrapping through the web page.

- **driver.findElement(By. name("name value"))**



4.XPath



If one has failed to identify an element by ID, class, or name, one would need to locate the element through its XML path. XPath is quite useful in locating dynamic elements on a webpage.

There are two types of location paths:

1. Absolute Path
2. Relative Path

Absolute XPath

```
<html>
<head>...</head>
<body>
...
<form id="loginForm">
<input name="name" type="text" value="First Name" />
<input name="name" type="text" value="Last Name" />
<input name="email" type="text" value="Business Email" />
<input name="password" type="password" />
<input name="continue" type="submit" value="Sign Me Up" />
</form>
</body>
</html>
```

Absolute XPath is the direct way to find the element. But the disadvantage of the absolute XPath is that if there are any changes made in the path of the element then that XPath fails. The key characteristic of XPath is that it begins with the single forward slash(/) ,which means you can select the element from the root node.

- **Absoulte XPath = html/body/form/input[3]**

Relative XPath

```
<html>
<head>...</head>
<body>
...
<form id="loginForm">
<input name="name" type="text" value="First Name" />
<input name="name" type="text" value="Last Name" />
<input name="email" type="text" value="Business Email" />
<input name="password" type="password" />
<input name="continue" type="submit" value="Sign Me Up" />
</form>
</body>
</html>
```

For Relative XPath, the path starts from the middle of the HTML DOM structure. It starts with the double forward slash (//), which means it can search the element anywhere at the webpage.

Common Format:: **//tagname[@attribute='value']**

- **Relative XPath =** **//input[@name='email']**

5.CSS Locator

A CSS Selector is a combination of an element selector and a value which identifies the web element within a web page. They are string representations of HTML tags, attributes, Id and Class. As such they are patterns that match against elements in a tree and are one of several technologies that can be used to select nodes in an XML document.

Types of CSS Selectors:

- ID
- Class
- Attribute