Оглавление

[03.03] - HTML Атрибуты

Анатомия HTML элемента.

1. **Открывающий тег (Opening tag)**: этот компонент HTML-структуры состоит из имени элемента, взятого в угловые скобки. Открывающий тег служит индикатором начала действия определенного элемента. В данном контексте он сигнализирует о начале нового абзаца, обозначенного тегом `<p>`.

2. **Закрывающий тег (Closing tag):** внешне похож на открывающий тег, но имеет дополнительную косую черту перед именем элемента. Этот тег указывает на место, где действие элемента завершается. В примере выше, он обозначает конец абзаца/параграфа. Отсутствие закрывающего тега — это одна из наиболее распространенных ошибок среди новичков в веб-разработке, и это может привести к непредсказуемым и неожиданным результатам.
    - Наглядный пример не закрытого тега, представлен по ссылке.

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">  <!-- Установка кодировки UTF-8 -->
    <title>Пример незакрытого тега</title>
</head>
<body>

    <p>Это первый абзац.</p>
    <p>Это второй абзац с <strong>важным текстом</p>
    <p>Это третий абзац.</p>
    <p>Это четвертый абзац.</p>


</body>
</html>
```

    На странице будет отображено:

- В этом примере отсутствует закрывающий тег: `</strong>` из второго абзаца. Из-за этой ошибки, текст "**Это третий абзац.**" и "**Это четвертый абзац.**" также будет отображен полужирным шрифтом. Следовательно, всё, что следует за открывающим тегом `<strong>` будет выделено, хотя таковым быть не должно.

1. **Контент (Content)**: этот термин описывает собственно содержимое элемента, которое в нашем случае является текстом. Контент располагается между открывающим и закрывающим тегами и представляет собой информацию, которую необходимо отобразить и именно контент мы и будем извлекать с помощью Selenium.
2. **Элемент(Element)**: открывающий тег, закрывающий тег и контент в совокупности формируют полноценный HTML-элемент. Элемент представляет собой базовую единицу структуры HTML-документа. Он служит для оформления или структурирования информации на веб-странице.

HTML тег действует как контейнер, где хранится контент. В этой роли, теги служат инструментами для организации и форматирования содержимого на веб-странице.

Одинарные и парные теги. В одинарных тегах таких как `<img>`, `<br>`, `<input>`, закрывающий тег отсутствует. Это потому, что такие теги не обрамляют дополнительный контент. С другой стороны, у парных тегов, таких как `<p>`, `<div>`, `<span>`, есть открывающий и закрывающий теги. Между ними находится сам контент.

Открывающий и закрывающий — две стороны одной медали: чтобы не было путаницы, открывающий и закрывающий теги — это не два разных объекта. Они являются различными аспектами одного и того же элемента в HTML. Они работают вместе, чтобы дать браузеру понять, как отобразить содержимое, заключенное между ними. Здесь ключевое понимание — они составляют единую структурную и функциональную единицу.

Элементы также могут иметь атрибуты, которые выглядят так:

Для успешной автоматизации с помощью Selenium, вам необходимо понимать структуру HTML-элементов, и здесь атрибуты играют ключевую роль. Это ваши "точки ориентиров" при извлечении информации с веб-страницы. Поэтому давайте разберёмся, как правильно "читать" атрибуты, чтобы потом успешно их "ловить" в дереве DOM.

1. **Пробел между атрибутом и именем элемента**: никаких исключений, пробел должен быть всегда. Если элемент уже имеет один или несколько атрибутов, пробел также должен располагаться между ними. Это ваш первый сигнал, что начинается новый атрибут, к которому можно "привязаться" при автоматизации.

2. **Имя атрибута и знак равенства**: после имени атрибута обязательно должен следовать знак равенства. Это важный момент, так как имя атрибута — это то, что вы будете использовать для поиска конкретного элемента на веб-странице. Если имя и знак равенства на месте — значит, вы на правильном пути.

```
# Пример того как мы будем искать элемент по двум атрибутам class и id
from selenium import webdriver
from selenium.webdriver.common.by import By

browser = webdriver.Chrome()
element = browser.find_element(By.CSS_SELECTOR, 'p.text#unique')
```

3. **Кавычки вокруг значения атрибута**: значение атрибута должно быть обрамлено кавычками — это непреложный закон HTML. Именно в этих кавычках находится информация, которую вы, возможно, захотите извлечь. Заметьте, что это могут быть как одинарные, так и двойные кавычки.

4. В HTML возможно применение нескольких классов к одному элементу, разделённых пробелами. Это позволяет применять к элементу сразу несколько стилей или упрощает выборку элементов при парсинге. Вот пример элемента с двойным классом:

```
<!-- Пример элемента с двойным классом -->
<p class="text highlight">Это абзац с двойным классом.</p>
```

Когда необходимо найти такой элемент с использованием Selenium, вы можете сделать это следующим образом:

```
# Поиск элемента с двойным классом
from selenium import webdriver
from selenium.webdriver.common.by import By

browser = webdriver.Chrome()
element = browser.find_element(By.CSS_SELECTOR, 'p.text.highlight')
```

Ключевые HTML Атрибуты

При автоматизации веб-страниц наша основная цель — извлечь нужную информацию из хаоса HTML-тегов. Атрибуты элементов играют роль своеобразных "якорей" или "указателей", которые помогают нам точно находить и идентифицировать необходимые данные. Знание основных атрибутов и понимание их роли в структуре документа критически важно для эффективной автоматизации. Давайте рассмотрим наиболее полезные атрибуты с точки зрения извлечения данных.

- ? **Атрибут `id`**

  Описание: Уникальный идентификатор элемента на всей странице может быть только один `id`.

  Так как `id` должен быть уникальным, это самый надежный способ найти конкретный, единственный в своем роде элемент. Если у нужного вам блока данных есть `id`, поиск значительно упрощается.

  ```
  <div id="user-profile">Информация о пользователе...</div>
  ```

  Пример поиска (Selenium)

  ```
  # Найти элемент по его уникальному id
  from selenium import webdriver
  from selenium.webdriver.common.by import By

  browser = webdriver.Chrome()
  user_div = browser.find_element(By.ID, 'user-profile')
  ```

- ?□□□ **Атрибут `class`**

  Описание: Задает один или несколько классов для элемента. Классы используются для группировки элементов (например, все карточки товаров могут иметь класс `product-card`) и для применения стилей CSS.

  Классы — один из самых частых способов найти **группу** однотипных элементов. Если вам нужно извлечь все товары, все новости или все комментарии, поиск по общему классу будет вашим основным инструментом.

  ```
  <div class="product-card">Карточка товара...</div>
  <div class="product-card">Другая карточка...</div>
  ```

  Пример поиска (Selenium):

  ```
  # Найти все элементы с классом 'product-card'
  from selenium import webdriver
  from selenium.webdriver.common.by import By
  ```

```
browser = webdriver.Chrome()
all_products = browser.find_elements(By.CLASS_NAME, 'product-card')

# Найти элемент с двумя классами 'product-card' и 'featured'
featured_product = browser.find_element(By.CSS_SELECTOR, '.product-
card.featured')
```

- ? **Атрибут `href`**

  Описание: Задает URL-адрес для ссылок (тег <a>).

  Крайне важен и не заменим для извлечения URL-адресов. Это основной способ собрать необходимые ссылки со страницы, например, для перехода на следующие страницы (пагинация) или для скачивания файлов.

  ```
  <a href="/catalog/item1">Ссылка на товар 1</a>
  ```

  Пример поиска/извлечения (Selenium):

  ```
  # Найти все ссылки и извлечь их URL
  from selenium import webdriver
  from selenium.webdriver.common.by import By

  browser = webdriver.Chrome()
  all_links = browser.find_elements(By.TAG_NAME, 'a')
  for link in all_links:
      url = link.get_attribute('href')
      print(url)
  ```

- ?□□□ **Атрибут `src`**

  Описание: Указывает источник (URL) для медиа-элементов, таких как изображения (<img>), скрипты (<script>), фреймы (<iframe>).

  Незаменим для извлечения URL-адресов изображений, видео или других встраиваемых ресурсов. Если вам нужно скачать все картинки со страницы, вы будете искать теги <img> и извлекать значение их атрибута `src`.

  ```
  <img src="/images/product1.jpg" alt="Фото товара 1">
  ```

- **Пример поиска/извлечения (Selenium):**

  ```
  # Найти все изображения и извлечь их URL
  from selenium import webdriver
  from selenium.webdriver.common.by import By

  browser = webdriver.Chrome()
  all_images = browser.find_elements(By.TAG_NAME, 'img')
  for img in all_images:
      img_url = img.get_attribute('src')
      print(img_url)
  ```

- **Атрибут `alt`**

  Описание: Задает альтернативный текст для изображений (тег <img>). Этот текст отображается, если изображение не может быть загружено, и используется программами чтения с экрана.

  Иногда `alt` текст содержит полезное описание изображения, которое можно

использовать как данные (например, название товара или краткое описание). Это может быть дополнительным источником информации.

```
<img src="/images/product1.jpg" alt="Смартфон 'Модель X' черный">
```

Пример извлечения (Selenium):

```
# Извлечь описание из атрибута alt
from selenium import webdriver
from selenium.webdriver.common.by import By

browser = webdriver.Chrome()
image = browser.find_element(By.TAG_NAME, 'img')
description = image.get_attribute('alt')
print(description) # Вывод: Смартфон 'Модель X' черный
```

- **Атрибут `title`**

  Описание: Предоставляет дополнительную информацию об элементе в виде всплывающей подсказки при наведении курсора мыши.

  Реже используется для ключевых данных, но иногда в `title` может содержаться полезная информация, недоступная в основном тексте элемента, например, полное название или уточнение.

```
<span title="Международный стандартный книжный номер">ISBN</span>
```

Пример извлечения (Selenium):

```
# Извлечь текст подсказки
from selenium import webdriver
from selenium.webdriver.common.by import By

browser = webdriver.Chrome()
span_tag = browser.find_element(By.TAG_NAME, 'span')
full_description = span_tag.get_attribute('title')
print(full_description) # Вывод: Международный стандартный книжный номер
```

- ? **Атрибут `style`**

  Описание: Используется для применения встроенных CSS-стилей непосредственно к элементу.

  Обычно напрямую не используется для автоматизации, но иногда стиль может указывать на состояние элемента (например, `style="display: none;"` означает, что элемент скрыт). Анализ стилей может помочь отфильтровать видимые элементы от скрытых, если это необходимо.

Пример HTML:

```
<div style="color: red; font-weight: bold;">Важное сообщение!</div>
<span style="display: none;">Скрытая информация</span>
```

Пример анализа (Selenium):

```
# Найти видимый элемент
from selenium import webdriver
from selenium.webdriver.common.by import By

browser = webdriver.Chrome()
div = browser.find_element(By.TAG_NAME, 'div')
```

```
style_attribute = div.get_attribute('style')
if 'display: none' not in style_attribute:
    print("Элемент видим:", div.text)
```

- ? **Атрибуты `data-*`**

  Описание: Пользовательские атрибуты данных, предназначенные для хранения дополнительной информации об элементе, которая не связана со стилями или стандартным поведением.

  Очень полезны! Веб-разработчики часто используют `data-*` атрибуты для хранения машиночитаемых данных, таких как ID товара, артикул, цена без форматирования, URL для AJAX-запроса и т.д. Это может быть золотой жилой для автоматизации, так как данные часто представлены в удобном для извлечения формате.

  ```
  <button class="add-to-cart" data-product-id="12345" data-
  price="99.99">Добавить в корзину</button>
  ```

  Пример извлечения (Selenium):

  ```
  # Извлечь данные из data-атрибутов
  from selenium import webdriver
  from selenium.webdriver.common.by import By

  browser = webdriver.Chrome()
  button = browser.find_element(By.CLASS_NAME, 'add-to-cart')
  product_id = button.get_attribute('data-product-id')
  price = button.get_attribute('data-price')
  print(f"ID товара: {product_id}, Цена: {price}") # Вывод: ID товара:
  12345, Цена: 99.99
  ```

Понимание этих атрибутов и умение их использовать при поиске элементов — ключ к созданию эффективных и надежных скриптов автоматизации.