Tổng hợp kiến thức Automation dành cho QA/Tester

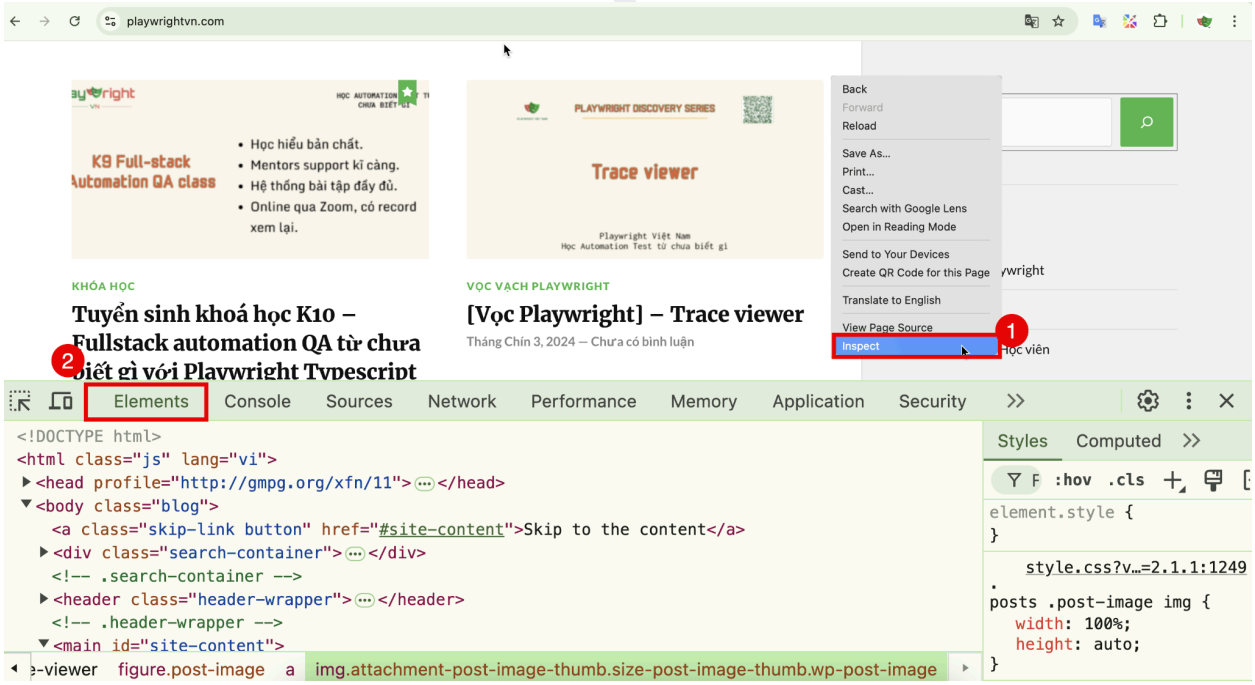
Lưu ý: Tài liệu này dành cho học viên khoá học Full-Stack automation QA với Playwright Typescript. Vui lòng không chia sẻ ra ngoài.
—

DOM

Khái niệm

- DOM là viết tắt của `Document Object Model`, là cách biểu diễn cấu trúc của trang web HTML.
- Là cách để mô tả cấu trúc và nội dung của một trang web sử dụng cú pháp của XML và HTML
- Để thấy được cấu trúc của trang web, bạn click chuột phải tại vùng trống, chọn inspect.

Sau khi Developer tools hiện ra, bạn click vào tab element để

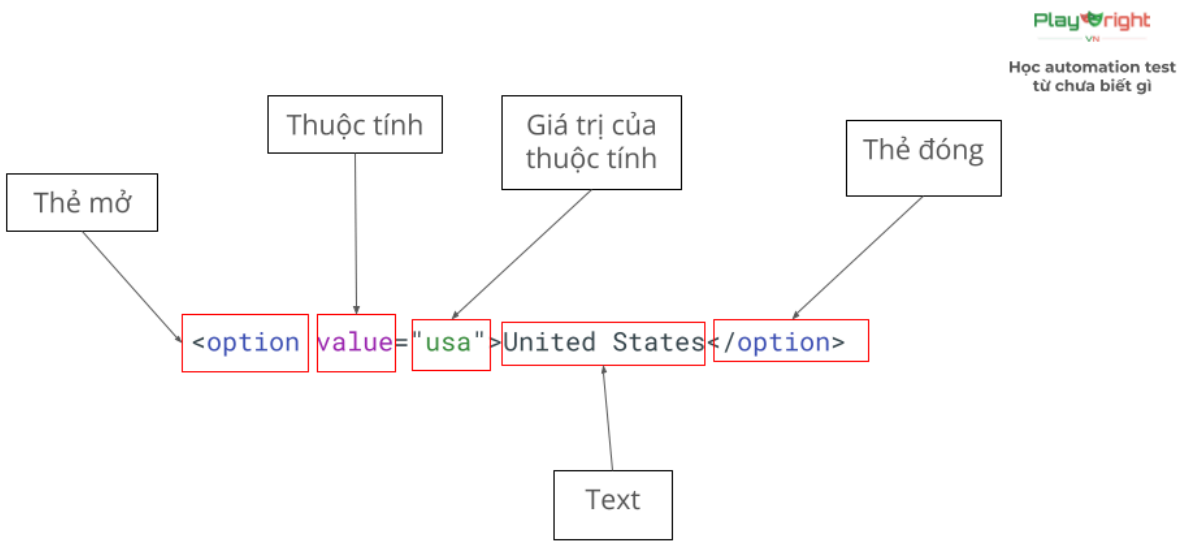


Một phần tử (còn được gọi là một node), bao gồm các thành phần chính như sau:

- Thẻ mở
- Thẻ đóng
- Text của node

- Thuộc tính
- Giá trị của thuộc tính

Hiểu được các thành phần cơ bản này, sẽ giúp chúng ta tìm kiếm phần tử trên trang một cách hiệu quả hơn.



Các loại thẻ

- Thẻ mở: `<div>`
- Thẻ đóng: `</div>`
- Thẻ tự đóng `<img/>`

Các thẻ HTML thường gặp

- Thẻ `<html></html>`: là thẻ chứa toàn bộ document của trang web.
- Thẻ `<head></head>`: chứa phần metadata, những dữ liệu liên quan đến tiêu đề trang web, icon của trang web, mô tả trang web, để cho các bot của Google, Facebook đọc.
- Thẻ `<body></body>`: chứa nội dung trang web, là phần mà người dùng nhìn thấy. Chúng ta sẽ làm việc chính với các nội dung bên trong thẻ này.

```
<!DOCTYPE html>
<html lang="en">
 ▶ <head> ... </head>
••• ▶ <body> ... </body> == $0
</html>
```

- Thẻ (viết tắt của divide): dùng để chia các khối trong trang web

```
<div class="search-container"></div>
```

- Thẻ <h1></h1> đến <h6></h6> (viết tắt của heading): dùng để tạo ra các header phân cấp theo thứ tự từ lớn đến bé.

```
<h1></h1>
<h2></h2>
<h3></h3>
<h4></h4>
<h5></h5>
<h6></h6>
```

Heading 1

Heading 2

Heading 3

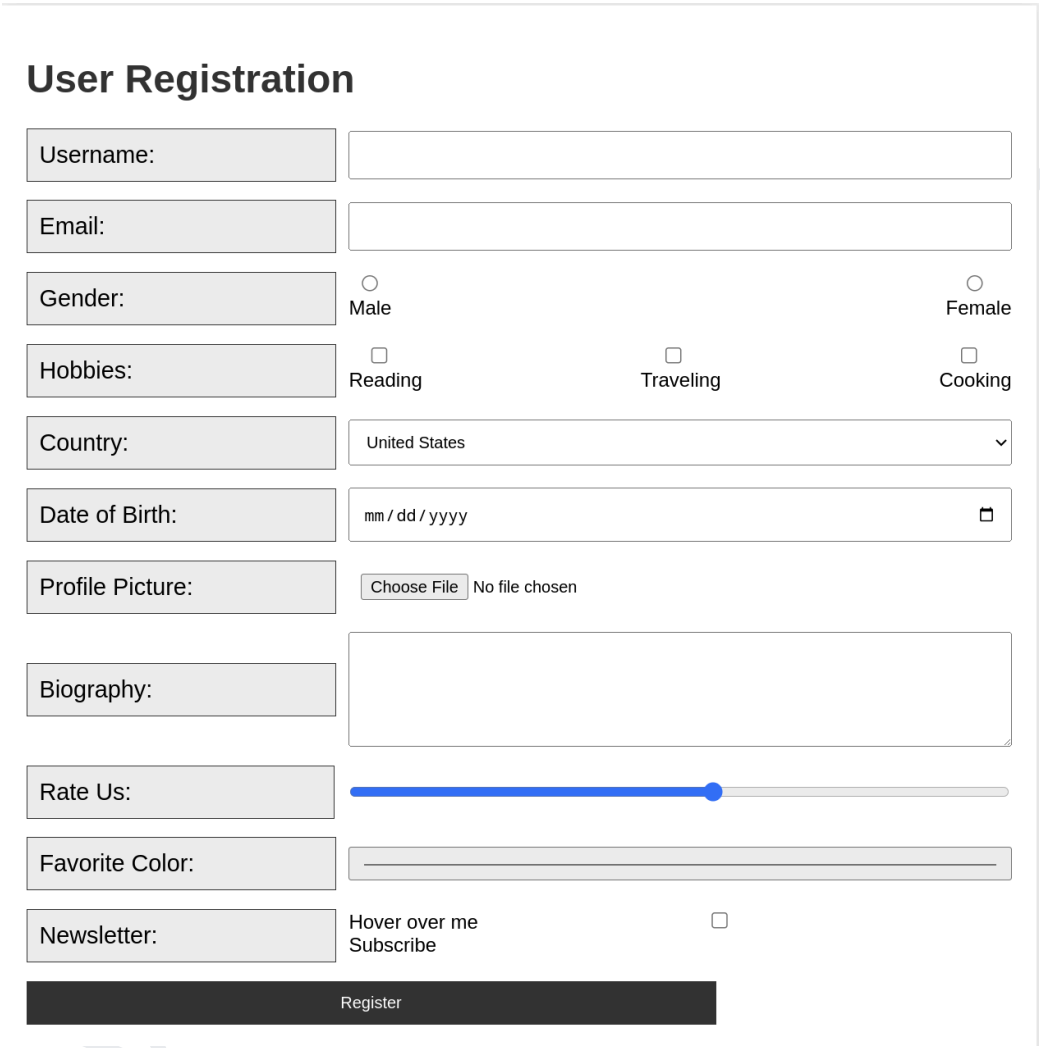
Heading 4

Heading 5

Heading 6

- Thẻ <form></form>: dùng để chứa một form thông tin.

```
<form id="registrationForm">
</form>
```



- Thẻ input text: thẻ dùng để nhập các dữ liệu dạng văn bản ngắn như username, phone number, address,...

```
<input type="text" id="username" name="username"
required=""/>: hiển thị ô input
```

- Thẻ input radio button: Hiển thị lựa chọn dạng chỉ chọn 1 lựa chọn duy nhất. Thường sử dụng trong ô chọn như giới tính.

```
<input type="radio" id="female" name="gender"
value="female">
```

Gender: ○ Male ○ Female ○ Not share

- Thẻ checkbox: Hiển thị lựa chọn dạng có thể chọn nhiều cùng lúc. Thường sử dụng trong ô chọn như sở thích,...

```
<input type="checkbox" id="reading" name="hobbies"
value="reading">
```

Hobbies: ☐ Reading ☐ Traveling ☐ Cooking

- Thẻ textarea: thẻ dùng để nhập các dữ liệu dạng văn bản dài, có nhiều dòng như một đoạn văn, mô tả sản phẩm,...

```
<textarea id="bio" name="bio" rows="4"></textarea>
```

Biography:

- Thẻ date picker: dùng để nhập các dữ liệu dạng ngày tháng năm. Format hiển thị của ô input thường đi theo hệ thống.

```
<input type="date" id="dob" name="dob">
```

Date of Birth: mm / dd / yyyy

- Thẻ slider: dùng để hiển thị các dữ liệu dạng %

```
<input type="range" id="rating" name="rating" min="1"
max="10">
```

- Thẻ list và dropdown: Hiển thị danh sách các lựa chọn.

```
<select id="country" name="country">
    <option value="usa">United States</option>
    <option value="canada">Canada</option>
    <option value="uk">United Kingdom</option>
    <option value="australia">Australia</option>
</select>
```

Country: United States

- Thẻ table: dùng để hiển thị dữ liệu dạng bảng. Trong thẻ table thường chứa các thẻ:
  - thead: phần đầu của table. Thường sẽ chứa tên các cột của bảng (dòng đầu tiên của bảng)
    - tr (table row): Chứa một dòng của bảng
    - th (table heading): chứa tên cột
  - tbody: phần thân, chứa nội dung của bảng
    - td: chứa nội dung của ô

```
<table id="userTable">
  <thead>
    <tr>
      <th>STT</th>
      <th>Username</th>
      <th>Email</th>
      <th>Information</th>
      <th>Actions</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td>1</td>
      <td>a</td>
      <td>a@gmail.com</td>
      <td>
        My info: Playwright Viet Nam
      </td>
      <td class="actions">
        <button onclick="editUser(0)">Edit</button>
        <button onclick="deleteUser(0)">Delete</button>
      </td>
    </tr>
  </tbody>
</table>
```

- Thẻ iframe: dùng để nhúng cả trang web vào trang hiện tại

```
<iframe src="https://www.example.com" title="Example
Iframe" width="100%" height="300px"></iframe>
```

DOM relation

DOM relation là mối quan hệ giữa các tag trong DOM.
Việc hiểu các mối quan hệ giúp bạn viết selector tốt hơn.

Mô hình hoá cây DOM

Mô hình hoá cây DOM là việc vẽ lại cây DOM trên các công cụ giống như MindMup, giúp bạn có một cái nhìn trực quan về cây DOM hơn.
Cách vẽ:

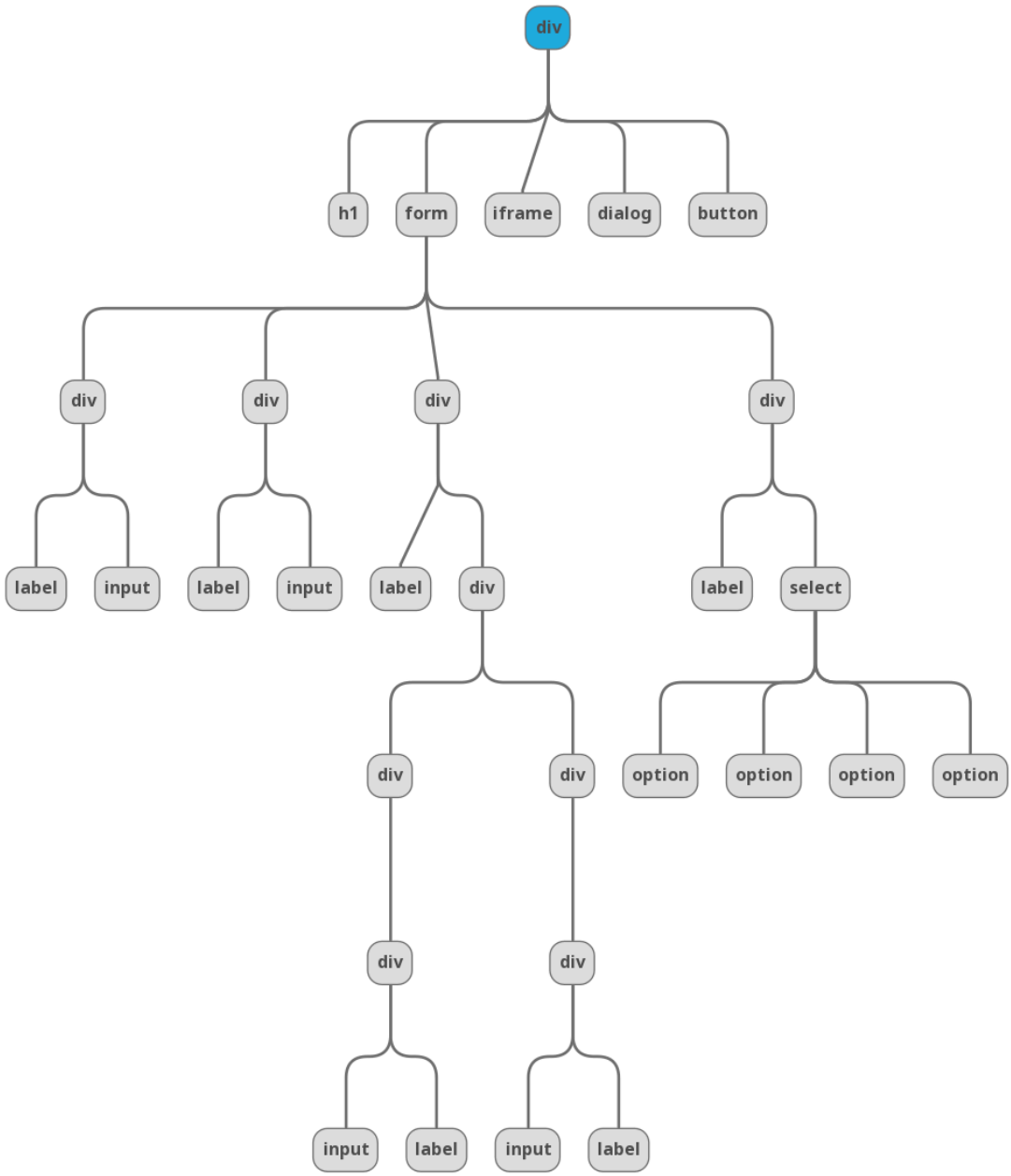
- Vẽ từ trên xuống dưới.
- Các node xếp ngay ngắn, thẳng hàng.

Ví dụ, với đoạn code html sau (xem trên GitHub):

```
<div class="container" id="ancestor">
    <h1 id="self">User Registration</h1>
    <form id="registrationForm">
        <div class="form-group" id="parent">
            <label for="username" id="preceding">Username:</label>
            <input type="text" id="username" name="username" required="">
        </div>
        <div class="form-group" id="child">
            <label for="email">Email:</label>
            <input type="email" id="email" name="email" required="">
        </div>
        <div class="form-group">
            <label for="gender">Gender:</label>
            <div style="width: 70%; display: flex; justify-content: space-between;">
                <div>
                    <input type="radio" id="male" name="gender" value="male">
                    <label nostyle="" for="male">Male</label>
                </div>
                <div>
                    <input type="radio" id="female" name="gender" value="female">
                    <label nostyle="" for="female">Female</label>
                </div>
            </div>
        </div>

        <div class="form-group">
            <label for="country">Country:</label>
            <select id="country" name="country">
                <option value="usa">United States</option>
                <option value="canada">Canada</option>
                <option value="uk">United Kingdom</option>
                <option value="australia">Australia</option>
            </select>
        </div>
    </form>
    <iframe src="https://www.example.com" title="Example Iframe" width="100%" height="300px"></iframe>
    <dialog id="dialog">This is a dialog</dialog>
    <button class="btn-home" onclick="goHome()">Trở về trang chủ</button>
</div>
```

Thì cây DOM sẽ như thế này:



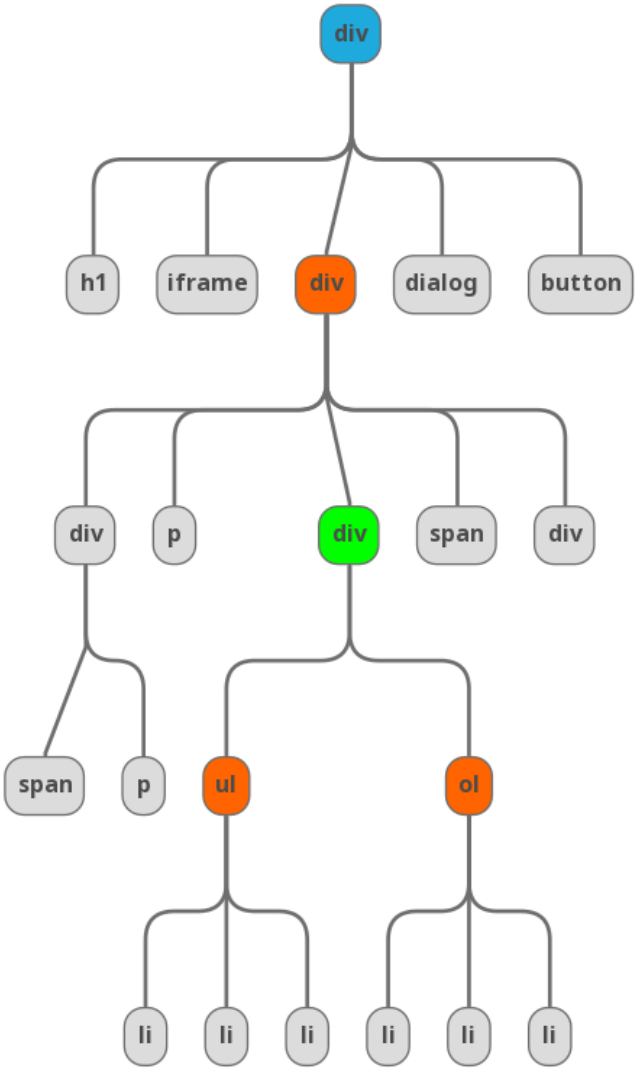
Quy ước màu sắc

Node gốc
Node cần chú ý
Node hiện tại

Mối quan hệ cha-con (parent-children)

Node cha là node **trực tiếp** phía trên của node hiện tại.
Node con là các node **trực tiếp** phía dưới của node hiện tại.

Trong hình dưới, thẻ cha là thẻ div, 2 thẻ con là thẻ ul và ol

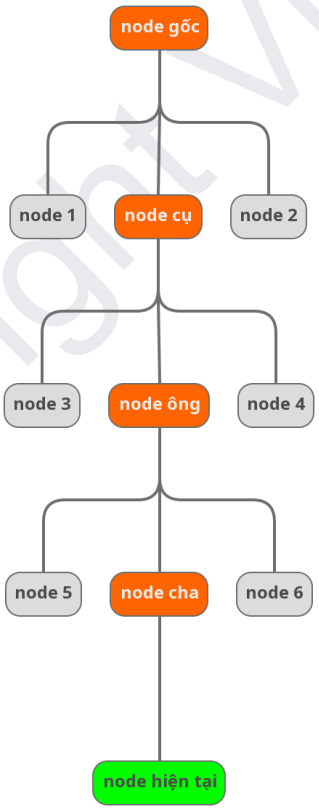


Mối quan hệ tổ tiên-hậu duệ (ancestor-descendant)

ancestor: tổ tiên

là các node:

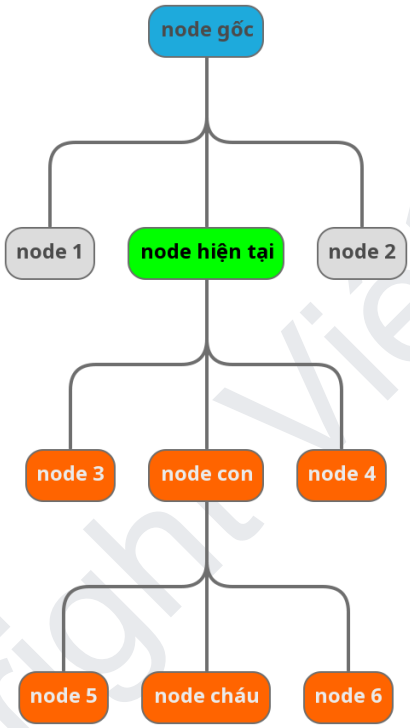
- (1) cha
- (2) cha của (1)
- (3) cha của (2)
- ...



descendant: hậu duệ

là các node con, cháu, chắt,...

Trong hình là các node: node3, node con, node 4, node 5, node cháu, node 6

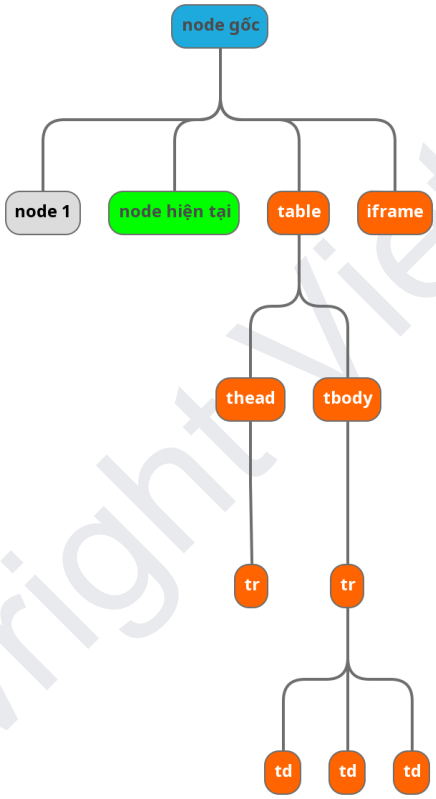


Mối quan hệ theo sau (following)

following: theo sau

Gồm các node ở phía bên tay phải của node hiện tại

Trong hình là các node: table, iframe, thead, tbody

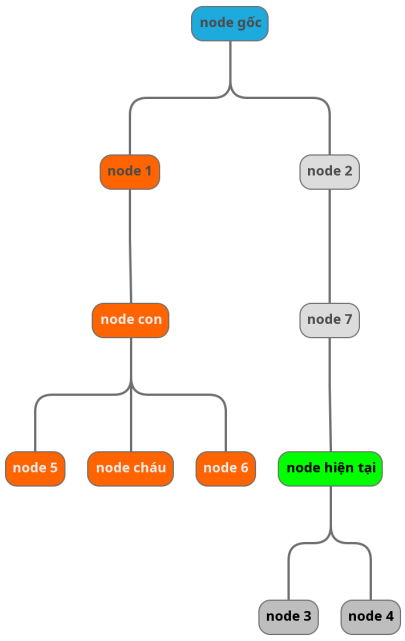


Mối quan hệ theo trước (preceding)

preceding: phía trước

Gồm các node ở phía bên tay trái của node hiện tại, **trừ các node ancestor**

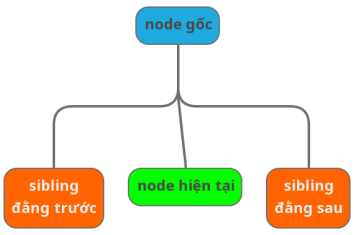
Trong hình là các node: div, h1, h2, h3



Mối quan hệ anh-em (sibling)

sibling: anh em

Là những phần tử cùng cấp và cùng cha

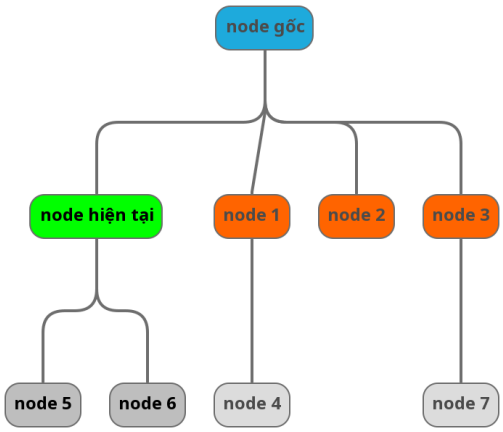


Kết hợp các mối quan hệ

Mối quan hệ anh-em-theo-sau (following-sibling)

following-sibling: anh em phía sau

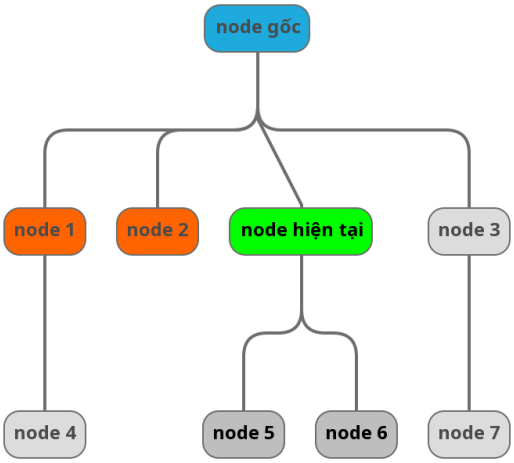
= following + sibling



Mối quan hệ anh-em-theo-trước (preceding-sibling)

preceding-sibling: anh em phía trước

= preceding + sibling



Selector

Khái niệm

Selector là một chuỗi hoặc biểu thức được sử dụng để xác định một hoặc nhiều phần tử HTML trên một trang web.
Selectors là thành phần quan trọng trong việc thao tác với DOM (Document Object Model) và thường được sử dụng trong CSS, JavaScript, và các công cụ kiểm thử tự động.
Nói một cách đơn giản, selector giúp bạn "chọn" các phần tử mà bạn muốn tác động đến (ví dụ: lấy nội dung, click vào chúng,...).

Các loại selector

- CSS Selectors
- XPath Selectors
- Playwright Selectors (chỉ có ở Playwright)

XPath selector

XPath (XML Path Language) là một ngôn ngữ truy vấn được sử dụng để điều hướng qua các phần tử và thuộc tính trong một tài liệu XML (Extensible Markup Language).
Do HTML có cấu trúc tương tự XML, XPath cũng có thể được sử dụng để chọn các phần tử HTML trên một trang web.

Các loại XPath selector

- XPath tuyệt đối (Absolute XPath)
- XPath tương đối (Relative XPath)

XPath tuyệt đối

- Khái niệm:
  - XPath tuyệt đối là một đường dẫn đầy đủ từ node gốc của tài liệu HTML đến phần tử bạn muốn chọn.
  - Nó bắt đầu bằng dấu / (slash) và chỉ ra đường dẫn chính xác đến phần tử, đi qua tất cả các node trung gian.
- Ví dụ
  Giả sử bạn có đoạn HTML sau:

```
<html>
    <head>
  <title>My Webpage</title>
    </head>
```

```
    <body>
        <div id="content">
            <h1>Welcome!</h1>
            <p>This is a paragraph.</p>
        </div>
    </body>
</html>
```

Để chọn thẻ <h1> bằng XPath tuyệt đối, bạn sẽ sử dụng:

```
/html/body/div[@id='content']/h1
```

- Ưu điểm:
  - Đảm bảo chọn đúng phần tử (nếu cấu trúc HTML không thay đổi).
- Nhược điểm:
  - Dễ bị fail: Chỉ một thay đổi nhỏ trong cấu trúc HTML (ví dụ: thêm một thẻ div trung gian) cũng có thể làm cho XPath tuyệt đối không còn hợp lệ.
  - Khó đọc và bảo trì: XPath tuyệt đối thường dài dòng và khó hiểu.
  - Không linh hoạt: Không thể thích ứng với các thay đổi nhỏ trên trang web.

XPath tương đối

- Khái niệm:
  - XPath tương đối không bắt đầu từ node gốc. Nó bắt đầu từ một node bất kỳ trong tài liệu và sử dụng các trục (axes) và hàm XPath để điều hướng đến phần tử bạn muốn chọn.
  - XPath tương đối bắt đầu bằng `//` (double slash).

- Ví dụ:
  - Sử dụng lại đoạn HTML ở trên, để chọn thẻ <h1> bằng XPath tương đối, bạn có thể sử dụng:

```
//div[@id='content']/h1
```

- Ưu điểm:
  - Ổn định hơn: Ít bị ảnh hưởng bởi các thay đổi nhỏ trong cấu trúc HTML.
  - Dễ đọc và bảo trì: XPath tương đối thường ngắn gọn và dễ hiểu hơn.
  - Linh hoạt hơn: Dễ dàng thích ứng với các thay đổi nhỏ trên trang web.

- Nhược điểm:
  - Có thể chọn sai phần tử nếu có nhiều phần tử trùng khớp.
  - Cần cẩn thận để đảm bảo XPath đủ cụ thể để chọn đúng phần tử mong muốn.

XPath advance

Sử dụng wildcard (*)

- Wildcard dùng để khớp tất cả các thẻ.
- Ví dụ:
  Xét đoạn html sau:

```
<div class="menu">Món ngon Việt Nam</div>
<ul>
    <li class="menu-item" order="1">Phở</li>
    <li class="menu-item" order="2">Bánh Mỳ</li>
</ul>
```

- //div: khớp thẻ div
- //*: khớp cả thẻ div, thẻ ul và thẻ li

Sử dụng or và and

- `or` và `and` dùng để kết hợp giữa các điều kiện với nhau.
- `or` là điều kiện hoặc. <a> or <b> thì chỉ cần <a> đúng hoặc <b> đúng là điều kiện thoả mãn.
- and là điều kiện và. <a> and <b> thì cần cả <a> và <b> đúng thì điều kiện mới thoả mãn
  Xét đoạn HTML sau:

```
<div class="menu">Món ngon Việt Nam</div>
<ul>
    <li class="menu-item" order="1">Phở</li>
    <li class="menu-item" order="2">Bánh Mỳ</li>
</ul>
```

- `//*[@class='menu' or @class='menu-item']`: khớp thẻ div, li vì các thẻ này có chứa class menu hoặc chứa class menu-item
- `//*[@class='menu-item' and @order='1']`: khớp item “Phở”, vì chỉ có thẻ này chứa class menu-item và order=1
  Bạn cũng có thể sử dụng kết hợp cả or và and
- `//*[(@class='menu' or class='menu-item') and @order='1']`

Sử dụng hàm text()

- Hàm text() dùng để tìm phần tử theo nội dung text của phần tử. Text là giá trị nằm giữa thẻ mở và thẻ đóng.
  Xét ví dụ thẻ html sau:
  ```
  <div class="menu">Món ngon Việt Nam</div>
  ```
- Thì giá trị text sẽ là “Món ngon Việt Nam”.
- Ví dụ sử dụng hàm text để tìm kiếm thẻ html trên:
  ```
  //div[text() = ‘Món ngon Việt Nam’]
  ```

parent

- parent dùng để đi đến thẻ cha của

Sử dụng [index]

- Toán tử [index] trong XPath được sử dụng để chọn một node cụ thể trong một node-set, dựa trên vị trí của nó. Nó cho phép bạn truy cập đến một node tại một vị trí cụ thể trong danh sách các node được chọn.
- Cú pháp: `//<tên_thẻ>[vị_trí] hoặc <đường_dẫn_xpath>[vị_trí]`
- Lưu ý quan trọng:
  - XPath indexes bắt đầu từ 1, không phải 0 như trong JavaScript.
  - [index] luôn được đặt ở cuối đường dẫn XPath (hoặc sau một bước trong đường dẫn).
- Ví dụ:
  Giả sử bạn có đoạn HTML sau:
  ```
  <ul>
      <li>Item 1</li>
      <li>Item 2</li>
      <li>Item 3</li>
      <li>Item 4</li>
  </ul>
  ```
  - `//li[1]`: Chọn phần tử <li> đầu tiên trong danh sách. Kết quả sẽ là Item 1.
  - `//li[2]`: Chọn phần tử <li> thứ hai trong danh sách. Kết quả sẽ là Item 2.

Element as attribute

- //tr[td[text()=’abc’]

Axes function

contains

- Mục đích: Chọn các node có thuộc tính hoặc text chứa một chuỗi con cụ thể.
- Cú pháp: `//*[contains(text(), 'chuỗi con')]` hoặc `//*[contains(@attribute, 'chuỗi con')]`
- Ví dụ (website hiển thị thời gian hiện tại):

  Giả sử bạn có một website hiển thị thời gian hiện tại, và thời gian được cập nhật mỗi khi bạn refresh trang. Đoạn HTML có thể như sau:

```
<div id="current-time">
    The current time is: 10:30:45 AM
</div>
```

  Bạn có thể sử dụng `contains()` để kiểm tra xem div `current-time` có chứa chuỗi "AM" hay không (để xác định xem thời gian có phải là buổi sáng hay không):

  xpath = `//div[@id='current-time'][contains(text(), 'AM')]`

  Giải thích: XPath này sẽ chọn div có `id` là `current-time` và có text chứa chuỗi "AM". Nếu bạn refresh trang và thời gian chuyển sang PM, XPath này sẽ không còn chọn node nào.

starts-with

- Mục đích: Chọn các node có thuộc tính hoặc text bắt đầu bằng một chuỗi cụ thể.
- Cú pháp: `//*[starts-with(text(), 'chuỗi bắt đầu')]` hoặc `//*[starts-with(@attribute, 'chuỗi bắt đầu')]`
- Ví dụ:

  Giả sử bạn có danh sách các sản phẩm, và một số sản phẩm có mã bắt đầu bằng "PROD-":

```
<ul>
    <li><span class="product-code">PROD-123</span> Product A</li>
    <li><span class="product-code">PROD-456</span> Product B</li>
```

```
        <li><span class="product-code">OTHER-789</span> Product
    C</li>
    </ul>
```

Bạn có thể sử dụng starts-with() để chọn tất cả các span có product-code bắt đầu bằng "PROD-":
xpath = `//span[starts-with(text(), 'PROD-')]`

position

- Mục đích: Trả về vị trí của node hiện tại trong tập hợp các node được chọn. Thường được sử dụng để chọn các node cụ thể dựa trên vị trí của chúng.
- Ví dụ (với table):
  Giả sử bạn có một bảng HTML như sau:

```
<table>
<thead>
    <tr>
        <th>Name</th>
        <th>Age</th>
    </tr>
</thead>
<tbody>
    <tr>
```

```
            <td>Alice</td>
            <td>25</td>
        </tr>
        <tr>
            <td>Bob</td>
            <td>30</td>
        </tr>
        <tr>
            <td>Charlie</td>
            <td>22</td>
        </tr>
    </tbody>
</table>
```

- `//tr[position() = 1]`: Chọn hàng đầu tiên (<tr>) trong bảng.
    - Lưu ý: Trong XPath, index bắt đầu từ 1, không phải 0 như trong JavaScript.
- `//tbody/tr[position() = 2]`: Chọn hàng thứ hai trong tbody (Bob).

last

- Mục đích: Trả về vị trí của node cuối cùng trong tập hợp các node được chọn.
- Ví dụ (tiếp tục với table trên):
    - `//tbody/tr[position() = last()]`: Chọn hàng cuối cùng trong tbody (Charlie).

count

- Mục đích: Đếm số lượng node trong một node-set.
- Ví dụ (bảng có 6 rows):
  Giả sử bạn có một bảng HTML và bạn muốn kiểm tra xem nó có đúng 6 hàng (bao gồm cả header) hay không:

```
<table>
    <thead>
        <tr><th>Header 1</th><th>Header 2</th></tr>
    </thead>
    <tbody>
        <tr><td>Row 1, Col 1</td><td>Row 1, Col 2</td></tr>
        <tr><td>Row 2, Col 1</td><td>Row 2, Col 2</td></tr>
```

```
            <tr><td>Row 3, Col 1</td><td>Row 3, Col 2</td></tr>
            <tr><td>Row 4, Col 1</td><td>Row 4, Col 2</td></tr>
            <tr><td>Row 5, Col 1</td><td>Row 5, Col 2</td></tr>
        </tbody>
    </table>
```

- `//table[count(.//tr) = 6]`: Chọn bảng chỉ khi nó có đúng 6 hàng (bao gồm cả thead).
- Giải thích:
  - `.//tr`: Chọn tất cả các thẻ tr là con cháu (descendant) của node hiện tại (trong trường hợp này là table).
  - `count(.//tr)`: Đếm số lượng thẻ tr được chọn.
  - `//table[count(.//tr) = 6]`: Chỉ chọn thẻ table nếu số lượng thẻ tr bên trong nó bằng 6.

Ignore cases and space

Ignore space - normalize-space()

Ignore case - translate

- //p[translate(text(), 'ABC…Z', 'abc…z')]
- Lưu ý về việc thay tương ứng vị trí
- Ví dụ có button UPPERCASE, dùng translate vẫn search ok

Kết hợp lại

normalize-space(translate())

string-length

- p[string-length(text()) < 30]

round and floor

- p[round(text())] = '53'
- p[floor(text()] = '54'

not

- p[not(@id, 'input-100)]

substring-before, substring-after

- p[substring-before(text(), ':') = '10am']

XPath axes methods

- XPath Axes cho phép bạn xác định mối quan hệ giữa các phần tử trong cây DOM.

Mối quan hệ giữa các phần tử trong DOM

Cú pháp chung cả xpath axes là: `//xpath/xpathAxesName::xpathAxesTag`

1. **parent**
    - Chọn phần tử cha trực tiếp của phần tử hiện tại
    - Cú pháp: `//tag/parent::parentTag`

    //div[@id='child']/parent::form
    -> Chọn phần tử cha `form` của phần tử `div` có id là 'child'.

2. **child**
    - Chọn tất cả phần tử con của phần tử hiện tại.
    - Có 2 cách
        - cách 1: `//tag//tagChild`
            - //div[@id='child']//label
        - cách 2: `//tag/child::childTag`
            - //div[@id='child']/child::label
3. **ancestor**
    - Chọn tất cả các phần tử tổ tiên (cha, ông, tổ tiên...) của phần tử hiện tại.
    - Cú pháp: `//tag/ancestor::ancestorTag`
    - Vd: //div[@id='child']/ancestor::*
4. **descendant**
    - Chọn tất cả các phần tử con và cháu (các phần tử con ở bất kỳ cấp độ nào) của phần tử hiện tại.
    - Cú pháp: `//tag/descendant::descendantTag`
    - Vd: //div[@id='child']/descendant::*

5. **sibling**
    - Đây không phải là một axes cụ thể nhưng được sử dụng để tham khảo các phần tử ngang hàng với phần tử hiện tại (bao gồm `following-sibling` và `preceding-sibling`).
6. **following**
    - Chọn tất cả các phần tử xuất hiện sau phần tử hiện tại trong DOM, không nhất thiết phải là anh chị em.
    - Cú pháp: `//tag/following::followingTag`
    - **Lưu ý: trừ các node con của node chính**
7. **preceding**
    - Ngược lại với following
    - Chọn tất cả các phần tử xuất hiện trước phần tử hiện tại trong DOM, không nhất thiết phải là anh chị em.
    - **Lưu ý: trừ các node ancestor**
8. **following-sibling**
    - Chọn tất cả các phần tử anh chị em đứng sau phần tử hiện tại trên cùng một cấp.
    - Cú pháp: `//tag/following-sibling::siblingTag`
9. **preceding-sibling**
    - Chọn tất cả các phần tử anh chị em đứng trước phần tử hiện tại trên cùng một cấp.
    - Cú pháp: `//tag/preceding-sibling::siblingTag`

CSS Selector

- CSS selectors là các mẫu được sử dụng để chọn các phần tử HTML dựa trên tên thẻ, class, ID, thuộc tính, và các mối quan hệ.
- Chúng thường được sử dụng để áp dụng các kiểu dáng CSS cho các phần tử, nhưng cũng có thể được sử dụng trong JavaScript để thao tác với DOM.

	CSS selector	Xpath selector
Tag	div	//div
id	#registrationForm	//form[@id="registrationForm"]
class	.form-group	//div[@class='form-group']
child	#parent > input	//div[@id='parent']/input
descendant	#ancestor div	//div[@id='ancestor']/descendant::div
combine	div, input	//div | //input
Adjacent sibling	#parent + div	//div[@id='parent']/following-sibling::*[1]
General sibling	#parent ~ div	//div[@id='parent']/following-sibling::*

Playwright selector

- Ưu điểm của Playwright selector là:
  - Tiện, nhanh
  - Cú pháp dễ sử dụng
- Nhược điểm:
  - Chỉ sử dụng trong Playwright
  - Không thử tìm trên browser được như xpath hay css selector.

1. page.getByRole()

- **Mô tả**: Dùng để tìm các phần tử dựa trên thuộc tính role của selector
- **Ví dụ**:

  HTML:
  ```
  <button role="button" aria-label="Submit">Submit</button>
  ```

  Playwright:

  ```
  await page.getByRole('button', { name: 'Submit' }).click();
  ```

- Ở đây, `getByRole` tìm button có role `button` và tên là `Submit`.

2. page.getByText()

- **Mô tả**: Tìm kiếm phần tử dựa trên nội dung văn bản.
- **Ví dụ**:

  HTML
  ```
  <div>Welcome to Playwright</div>
  ```

  Playwright:

  ```
  await page.getByText('Welcome to Playwright').click();
  ```

- Dòng này tìm và click vào phần tử có chứa văn bản "Welcome to Playwright".

3. page.getByLabel()

- **Mô tả**: Tìm selector dựa trên nội dung text của thẻ <label>.
- **Ví dụ**

  HTML
  ```
  <label for="username">Username</label>
  <input id="username" type="text" />
  ```

  Playwright:

  ```
  await page.getByLabel('Username').fill('user1');
  ```

- Phần tử input có label "Username" sẽ được tìm thấy và điền giá trị 'user1'.

4. page.getByPlaceholder()

- **Mô tả**: Dựa trên thuộc tính placeholder.
- **Ví dụ**

HTML
```
<input type="email" placeholder="Enter your email" />
```

Playwright:

```
await page.getByPlaceholder('Enter your email').fill('user@example.com');
```

5. page.getByAltText()

- **Mô tả**: Tìm kiếm phần tử, thường là hình ảnh, dựa trên văn bản thay thế (alt text).
- **Ví dụ**

HTML
```
<img src="logo.png" alt="Company Logo" />
```

Playwright:

```
await page.getByAltText('Company Logo').click();
```

- Dòng này sẽ tìm hình ảnh có alt text "Company Logo" và thực hiện click.

6. page.getByTitle()

- **Mô tả**: Lấy phần tử dựa trên thuộc tính title.
- **Ví dụ**

HTML
```
<button title="Close">X</button>
```

Playwright:

```
await page.getByTitle('Close').click();
```

- Phần tử có title là "Close" sẽ được tìm và click.

7. page.getByTestId()

- **Mô tả**: Tìm kiếm phần tử dựa trên thuộc tính `data-testid`, hoặc các thuộc tính khác có thể được config.
- **Ví dụ**

  HTML
  ```
  <button data-testid="submit-button">Submit</button>

  Playwright:

  await page.getByTestId('submit-button').click();
  ```

- Phần tử có `data-testid` là "submit-button" sẽ được tìm và nhấn vào.

- Note
  - `page.getByTestId()` trong Playwright được thiết kế đặc biệt để tìm các phần tử dựa trên thuộc tính `data-testid`. Mặc định, nó chỉ hoạt động với `data-testid`. Tuy nhiên, bạn có thể cấu hình Playwright để cho phép `page.getByTestId()` sử dụng các thuộc tính khác, bao gồm cả `id`.
  - Nếu bạn muốn sử dụng `page.getByTestId()` với thuộc tính `id`, bạn có thể cấu hình Playwright như sau:

    ```
    const { test, expect } = require('@playwright/test');

    test('example test', async ({ page }) => {
      // Cấu hình để sử dụng 'id' thay cho 'data-testid'
      await page.setTestIdAttribute('id');

      // Bây giờ bạn có thể sử dụng page.getByTestId với 'id'
      await page.getByTestId('submit-button').click();
    });
    ```