

Mastering XPath and CSS selectors



/ Mohammad Monfared

1. XPath Basics

- Syntax
- Absolute and Relative
- and / or
- Parents and Children
- Index
- Nested locators

Syntax:

'//' Means anywhere in the DOM

Expression should always be between brackets

// * [@attribute = 'value']

Tag

@name
text()

Values should always be between quotes

'@' should always be here except for text locators

Absolute XPATH

`/html/body/div[3]/div[2]/form/fieldset/input[9]`

Relational XPATH

`// input [name='Options']`

// tag [condition1] # Simple

// tag [condition1 and condition2] # 'AND' logic

// tag [condition1 or condition2] # 'OR' logic

// tag [(condition1 or condition2) and condition3] # 'AND' + 'OR'

// tag [condition1] /.. #Parent

// tag [condition1] / tag [condition2] # Child

// tag [condition1] // tag [condition2] # Search through children

`// tag [condition1] [index]` #Nth child (of its parent) with this locator

`(// tag [condition1]) [index]` #Nth element with this locator

`// tag [tag [condition1]]` # Nested

`// tag [.// tag [condition1]]` # Nested

2. XPath Functions

- contains()
- starts-with()
- position()
- last()
- count()
- normalize-space()
- translate()
- string-length()
- round()
- floor()
- not()
- substring-before()
- substring-after()

contains()

```
// * [ contains(text(), 'string')]
```

```
// * [ contains(text(), 'string') and @id='some-id']
```


starts-with()

```
// * [ starts-with(text(), 'string')]
```

```
<br>
... <label for="datalists">Start typing and it will automatically guess
    answer:</label> == $0
    <br>
    <input list="datalists" name="Options">
    ▶ <datalist id="datalists">...</datalist>
    <br>
    <br>
    <label for="favcolor">Select your favorite color:</label>
```

html body div.row div.main form fieldset label

```
/* [ starts-with ( text() , 'Start typing')]
```

1 of 1 ^ v Cancel

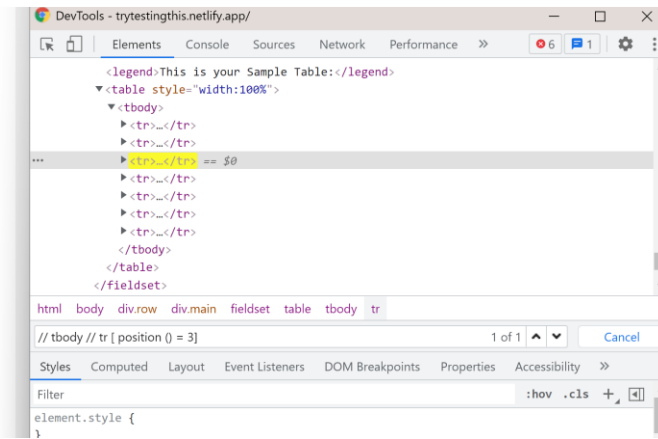
position()

// * [position() = 3] //starts from 1

Title description, Sep 2, 2017

This is your Sample Table:

Firstname	Lastname	Gender	Age	Occupation
561.4 x 49	Geller	F	27	Chef
Pheobe	Buffay	F	27	Singer
Joey	Tribbiani	M	28	Actor
Chanandler	Bong	O	28	Transponster
Ross	Geller	M	29	Paleontologist
Rachel	Green	F	27	Personal Shopper

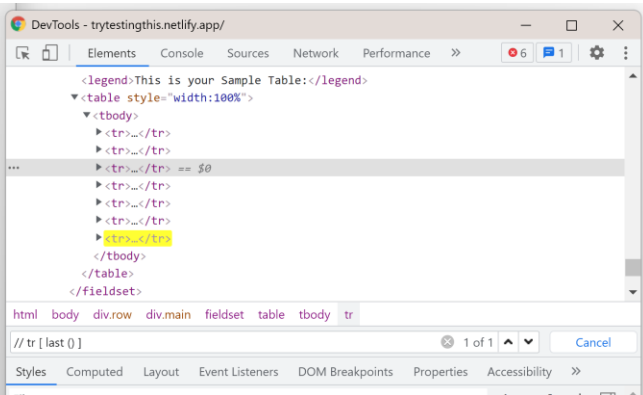


last()

// * [last()]

This is your Sample Table:

Firstname	Lastname	Gender	Age	Occupation
Monika	Geller	F	27	Chef
Pheobe	Buffay	F	27	Singer
Joey	Tribbiani	M	28	Actor
Chanandler	Bong	O	28	Transponster
tr 561,4 x 49	Geller	M	29	Paleontogist
Rachel	Green	F	27	Personal Shopper



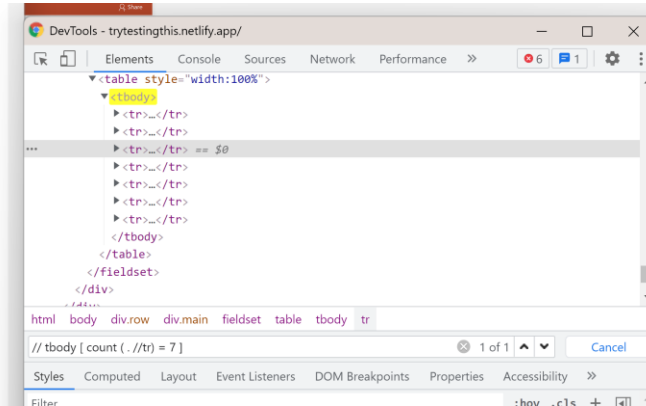
count()

```
// tbody [ count (./tr) = 7 ]
```

tbody 561.4 x 343

Table:

Firstname	Lastname	Gender	Age	Occupation
Monika	Geller	F	27	Chef
Pheobe	Buffay	F	27	Singer
Joey	Tribbiani	M	28	Actor
Chanandler	Bong	O	28	Transponster
Ross	Geller	M	29	Paleontologist
Rachel	Green	F	27	Personal Shopper



normalize-space()

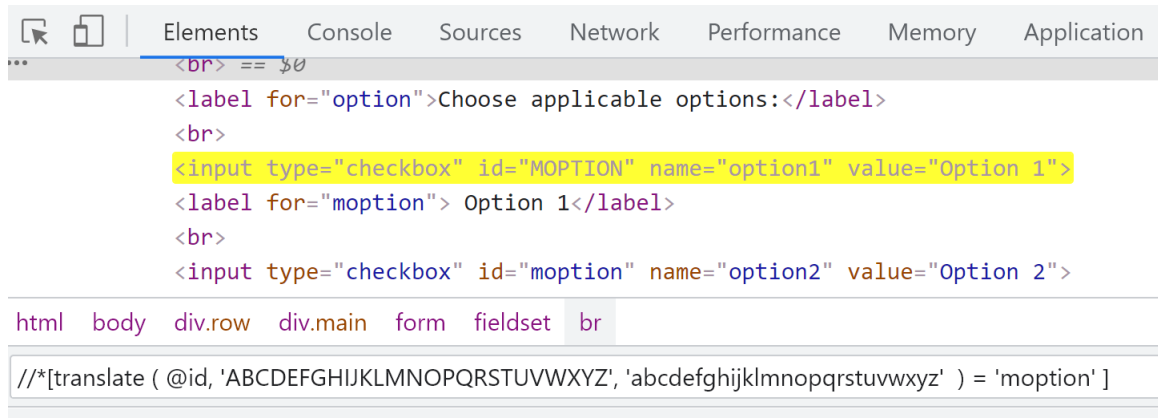
```
// *[ normalize-space ( text() ) = 'Option1' ]
```

```
// *[ normalize-space (@id) = 'Option1' ]
```

```
<br>
<label for="option">Choose applicable options:</label>
<br>
<input type="checkbox" id="moption" name="option1" value="Optio
.. <label for="moption"> Option 1</label> == $0
html body div.row div.main form fieldset label
/*[normalize-space ( text() ) = 'Option 1' ]
```

translate()

```
// *[ translate (text(), 'OPTION', 'option') = 'option1' ]
```



The screenshot shows a web browser's developer tools interface. The 'Elements' tab is selected, displaying the DOM tree. The tree structure is as follows:

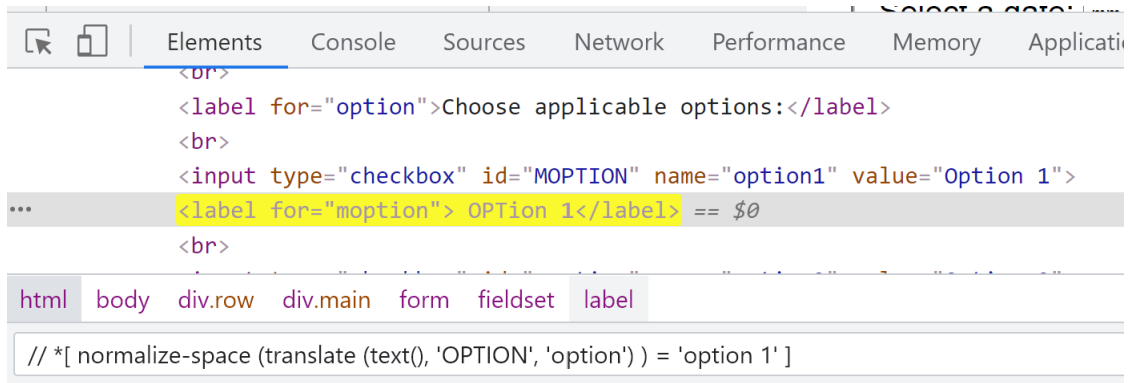
- html
 - body
 - div.row
 - div.main
 - form
 - fieldset
 - br (selected)

The selected `br` element's HTML source code is shown at the bottom:

```
/*[translate ( @id, 'ABCDEFGHIJKLMNOPQRSTUVWXYZ', 'abcdefghijklmnopqrstuvwxyz' ) = 'moption' ]
```

Combine normalize-space() and translate()

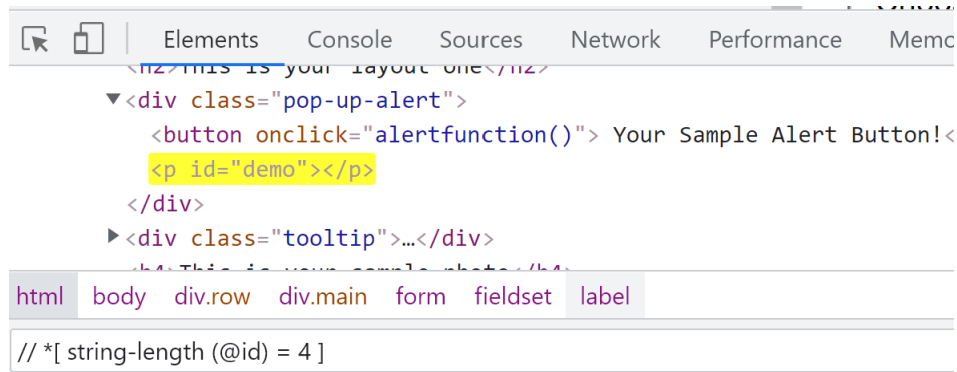
```
// *[ normalize-space (translate (text(), 'OPTION', 'option')) ] = 'option1' ]
```



string-length()

```
// *[ string-length (text()) > 30 ]
```

```
// *[ string-length (@id) = 4 ]
```



round()

```
// td [ text() = '53.76' ]
```

OR

```
// td [ round (text()) = '54']
```

floor()

```
// td [ text() = '53.76' ]
```

OR

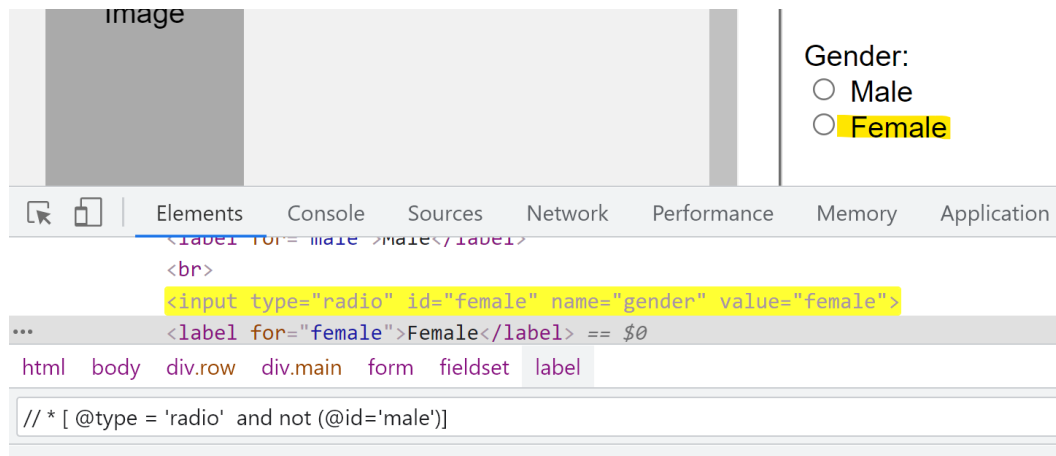
```
// td [ floor (text()) = '53']
```

not()

```
// * [ type() = 'radio' and @id='female']
```

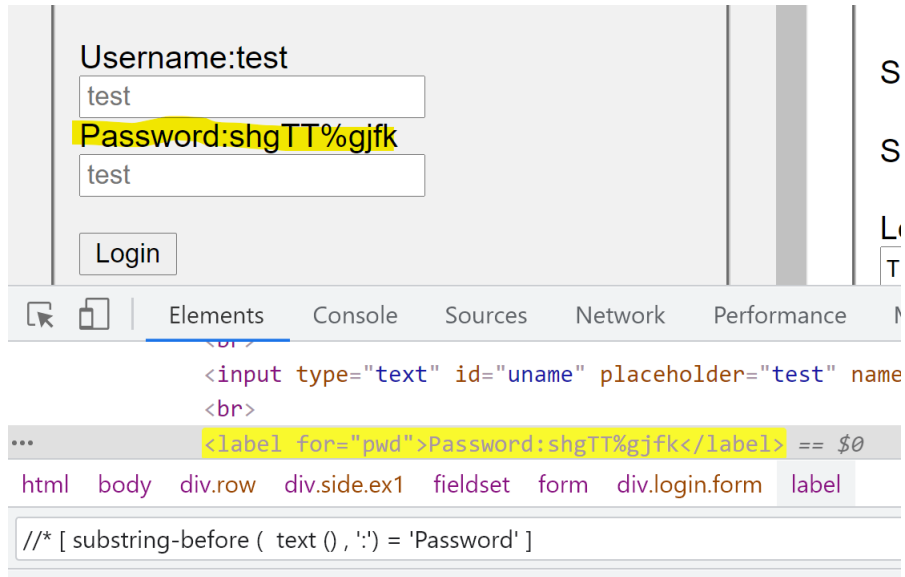
OR

```
// * [ type() = 'radio' and not (@id='male')]
```



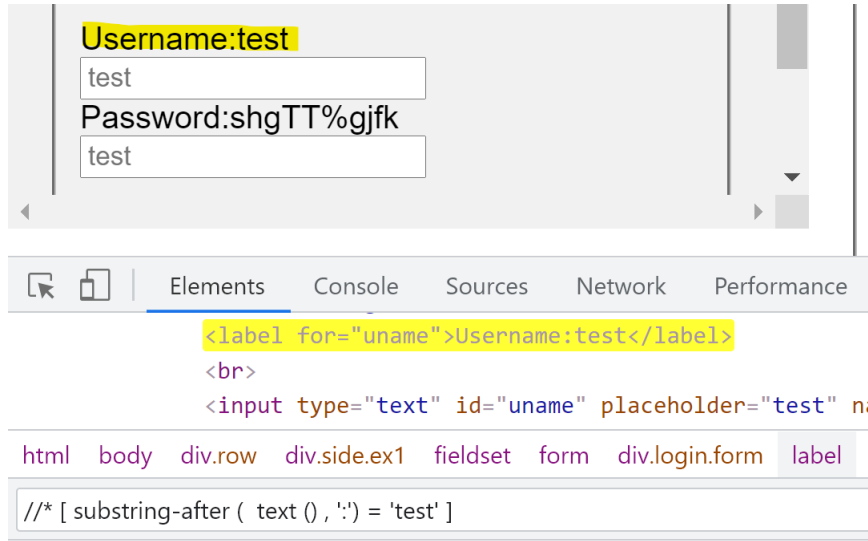
substring-before()

```
/** [ substring-before ( text () , ':' ) = 'Password' ]
```



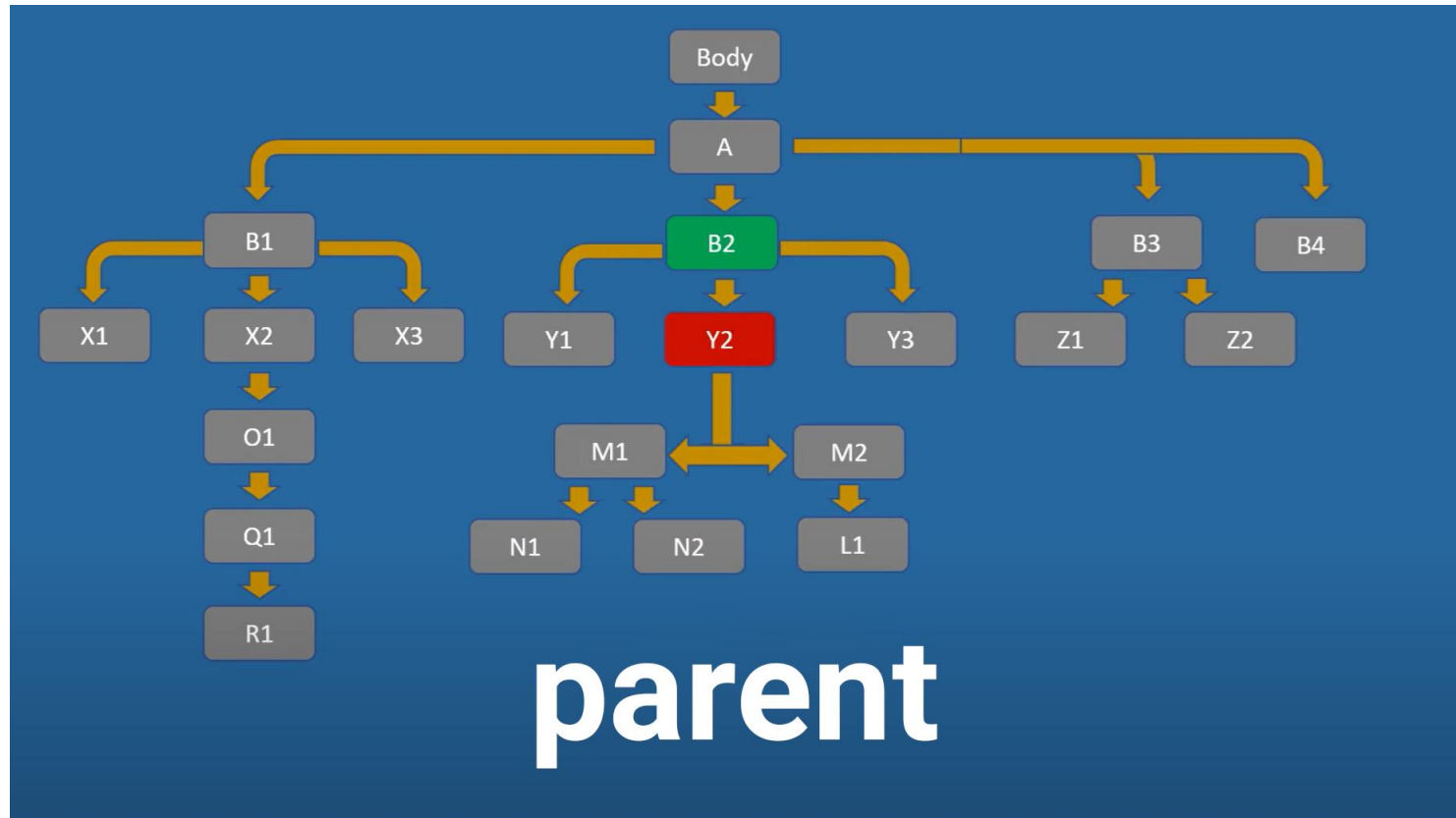
substring-after()

```
//* [ substring-after ( text () , ':' ) = 'Password' ]
```

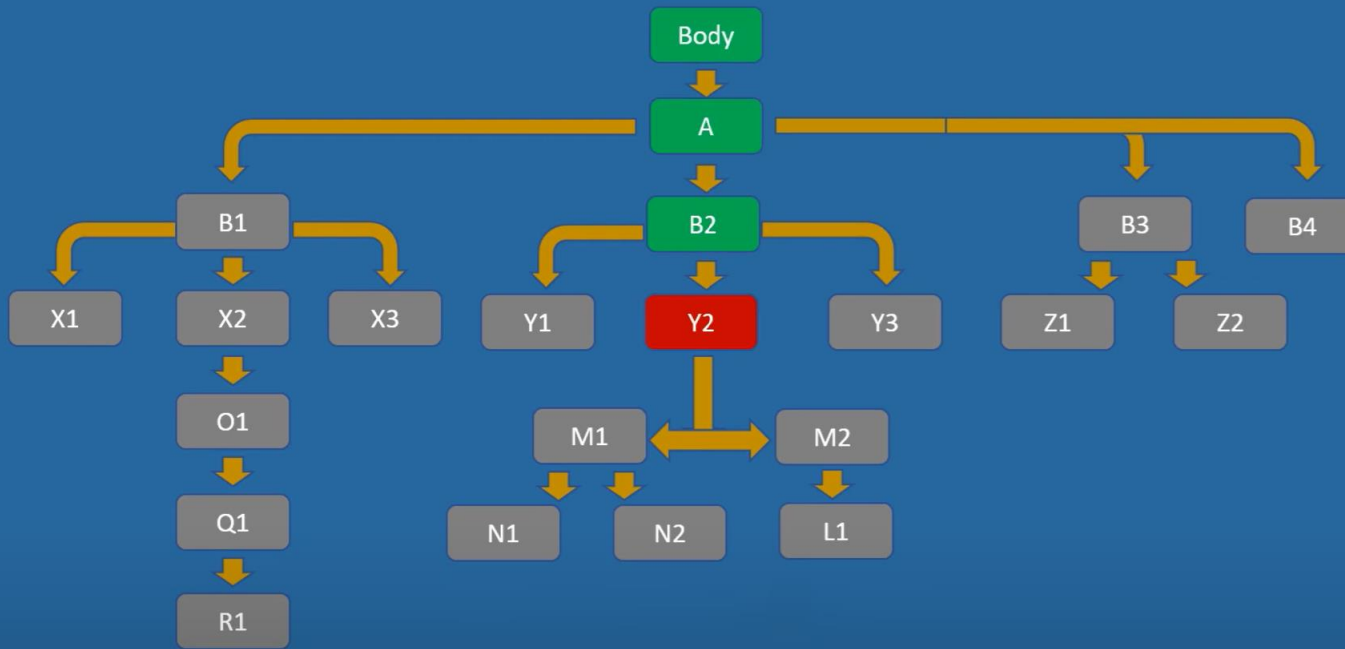


3. XPath Axes

- parent
- ancestor
- ancestor-or-self
- child
- descendent
- descendent-or-self
- following
- following-sibling
- preceding
- preceding-sibling

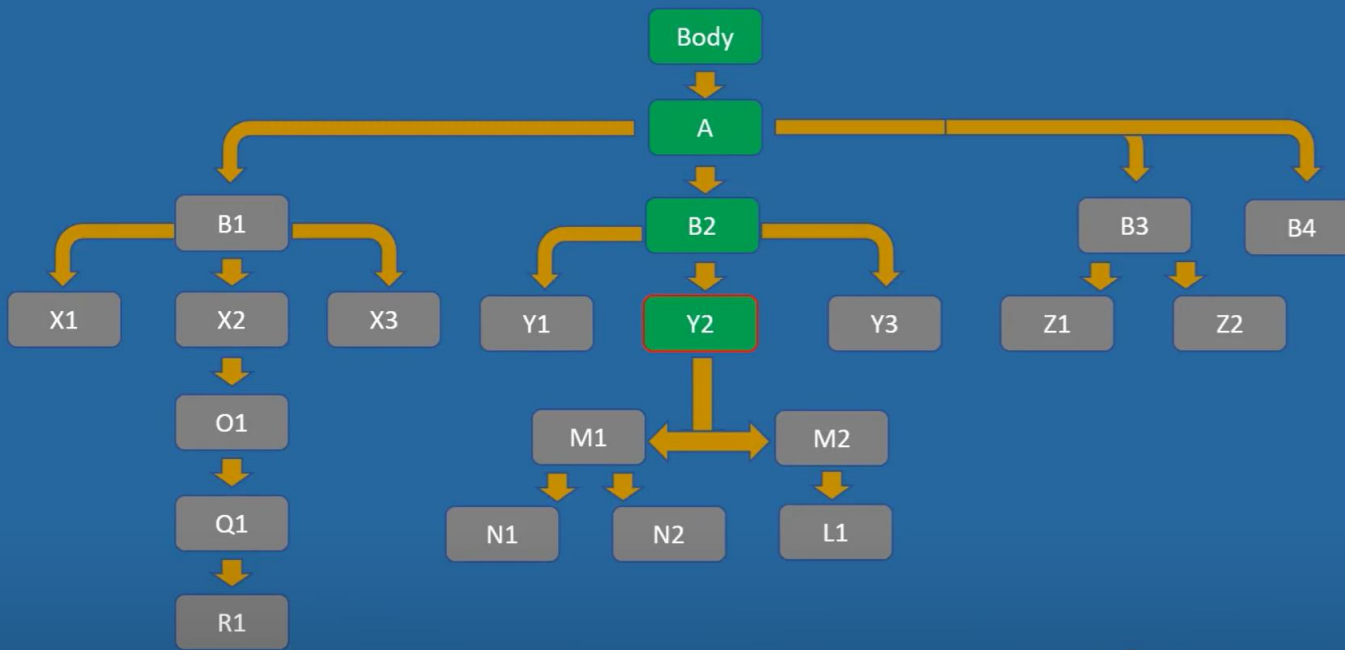


```
/* [@id='abc'] / parent :: *[@id='def']
```



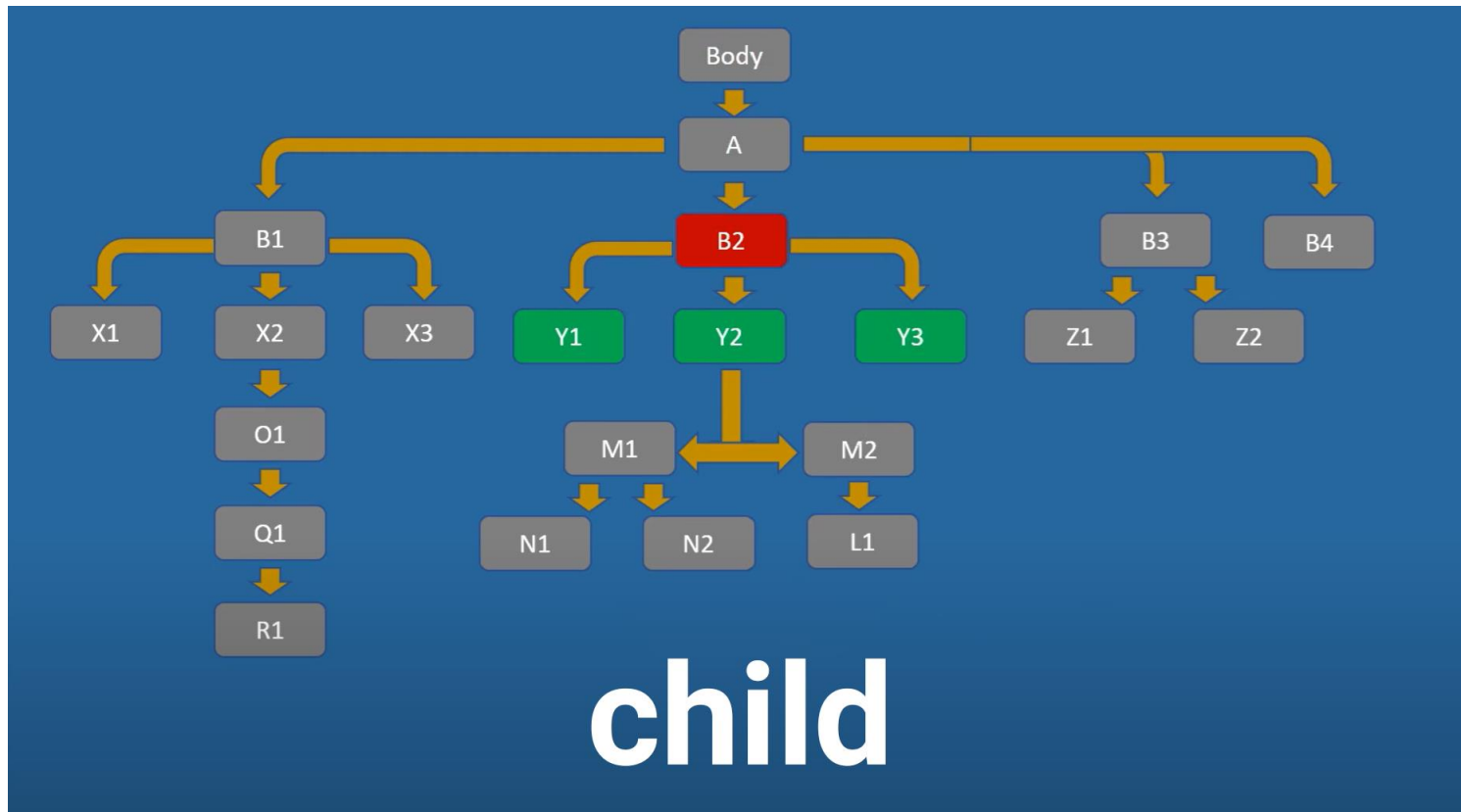
ancestor

`//* [@id='abc'] / ancestor :: *[@id='def']`

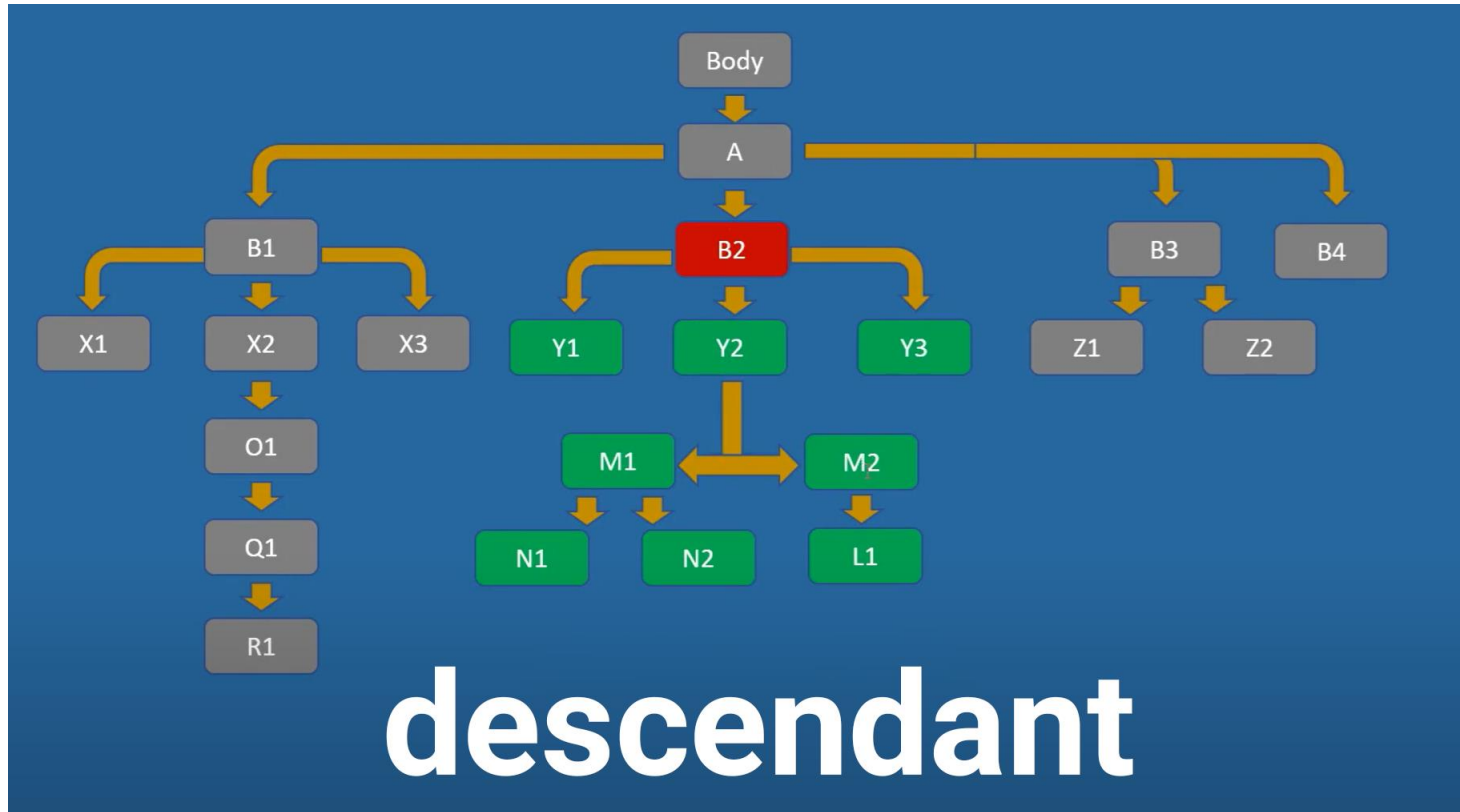


ancestor-or-self

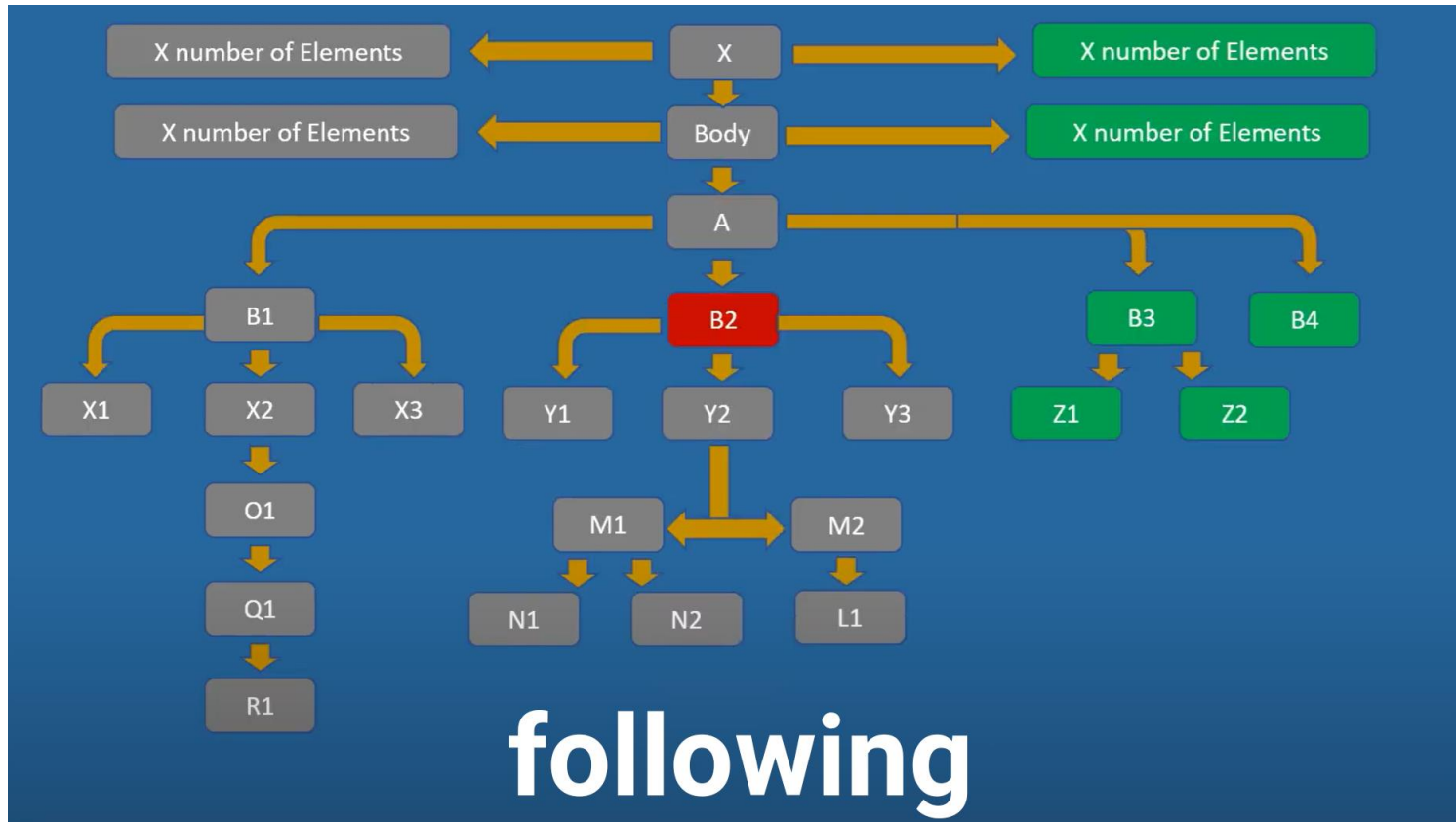
`//* [@id='abc'] / ancestor-or-self :: *[@id='def']`



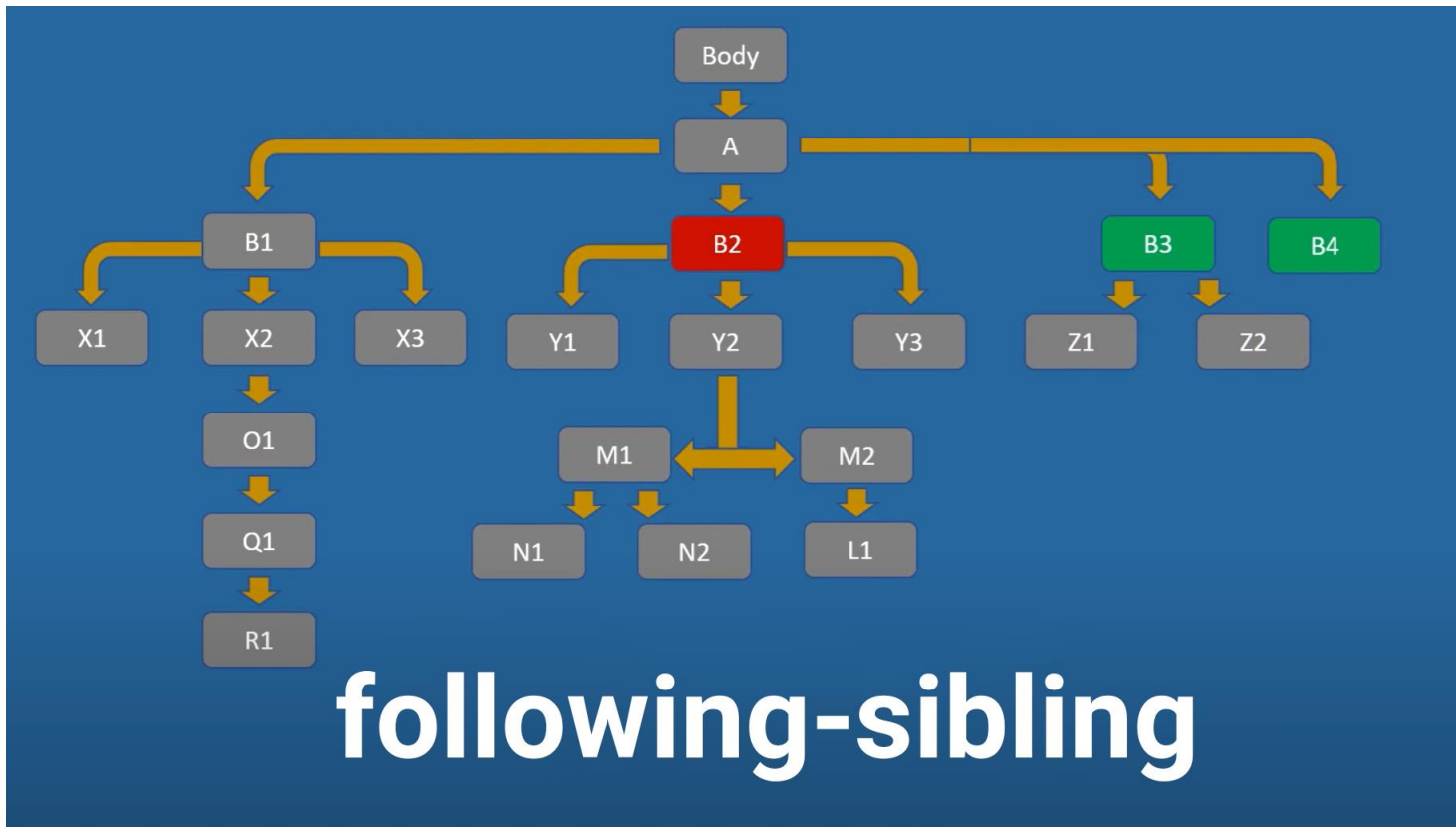
`//* [@id='abc'] / child :: *[@id='def']`



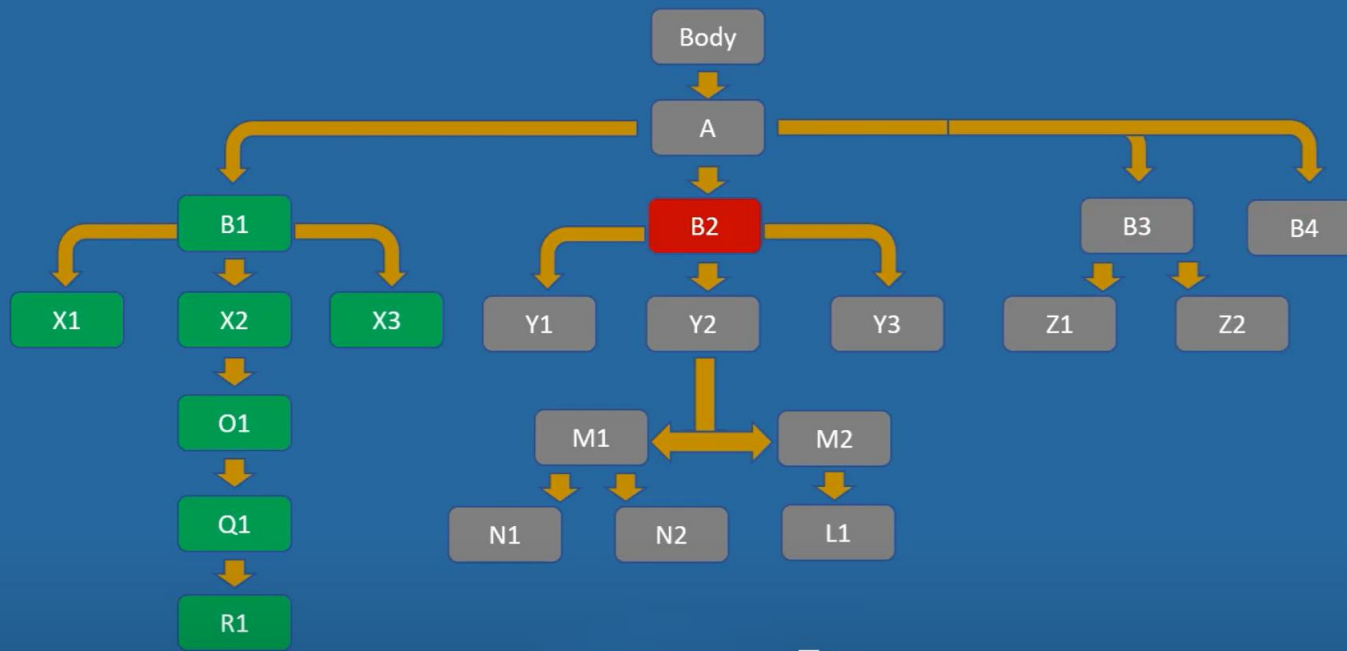
`//* [@id='abc'] / descendant :: *[@id='def']`



//* [@id='abc'] / following :: *[@id='def']

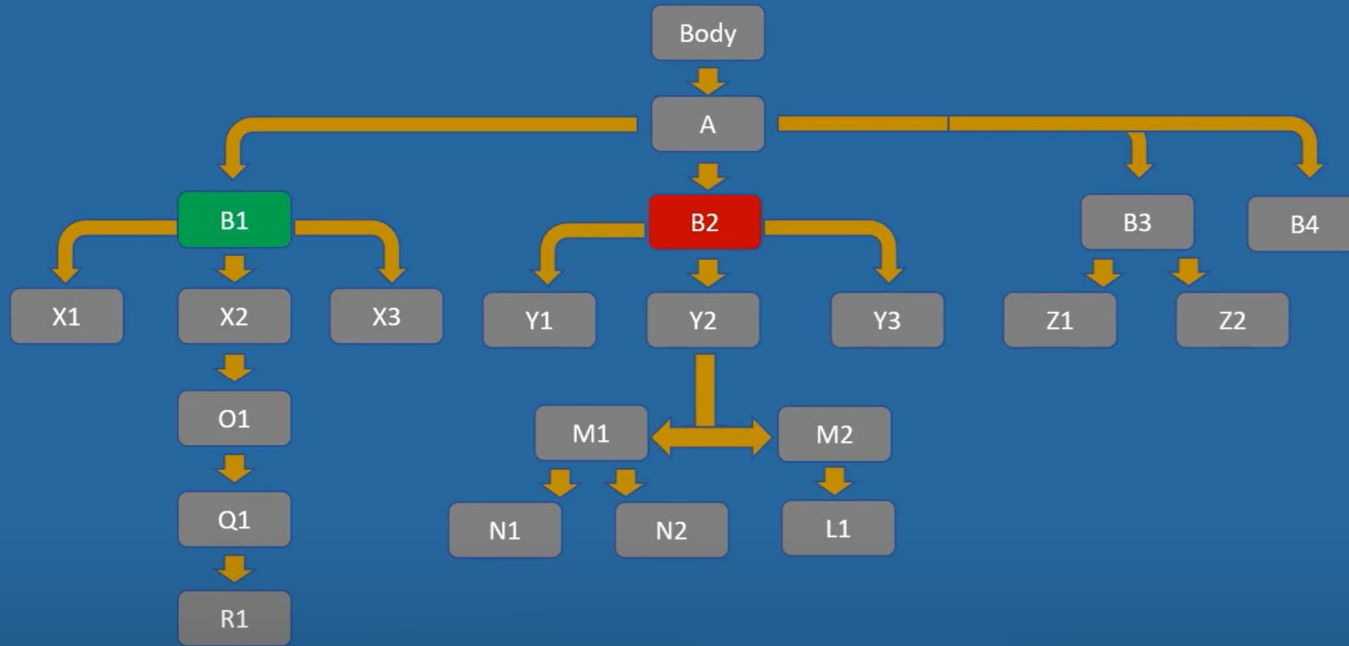


```
//* [@id='abc'] / following-sibling :: *[@id='def']
```



preceding

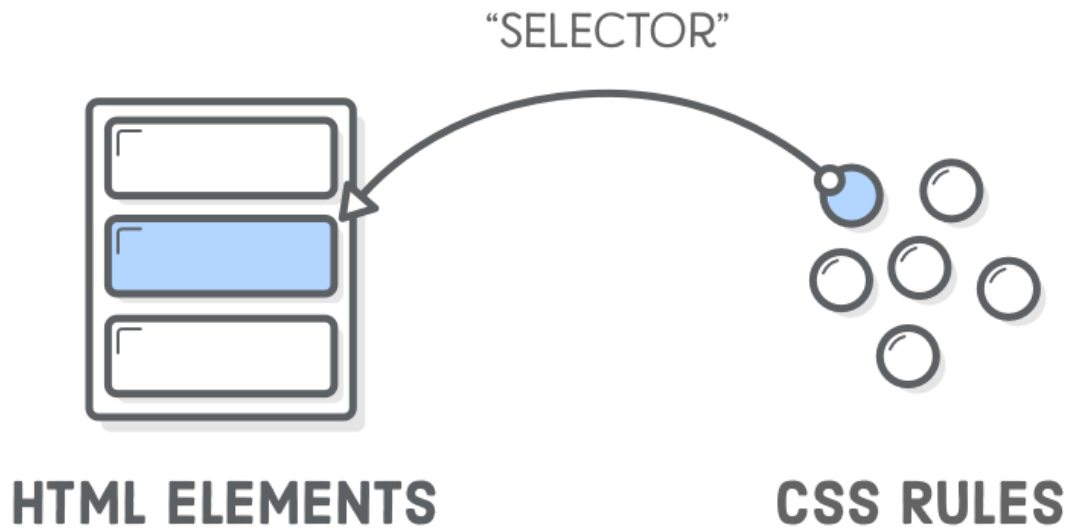
`//* [@id='abc'] / preceding :: *[@id='def']`



preceding-sibling

`//* [@id='abc'] / preceding-sibling :: *[@id='def']`

4. CSS Selectors



Advantages

1. CSS selectors typically offer better, faster, and more reliable performance than XPath in most web browsers.
2. All the JS-based test framework like Cypress use CSS selectors
3. It is necessary to use CSS locators when we are using the JS executor to locate elements in Selenium/Appium.
4. To perform useful functions within selectors, such as 'select all checked checkboxes' or 'select all visited links', we need to use CSS pseudo-functions.
5. An essential fundamental for frontend developers.

Basic Syntax

tag [AttrName = 'AttrValue']

Sample expressions:

input[id = 'fname']

button[value = 'submit']

ID & Class Selectors

tag **#** ID-value

input #fname

tag **.** Class-value

div . tooltip

Combination

tag # ID-value [AttrName = 'AttrValue']

tag # ID-value . Class-value

AND (for attributes of a element)

tag # ID-value [Attr1Name = 'Attr1Value'] [Attr2Name = 'Attr2Value']

OR (select any element of given expressions)

tag1. Class-value , tag2 . Class-value

Sub-string Match

<input type="checkbox" id="moption">

‘ ^ ’ Starts with *input[id^='mop']*

‘ \$ ’ Ends with *input[id\$='tion']*

‘ * ’ Partial text *input[id*='opt']*

Can be combined as well:

input[id='opt'][type\$='box']*

Child/Descendant

Direct child

tag # ID-value>tag # ID-value

Descendant (Child and sub-child)

tag # ID-value tag # ID-value

Next-sibling (Adjacent)

tag # ID-value+tag # ID-value

Pseudo Class - Child

First child

tag # ID-value **:first-child**

Last child

tag # ID-value **:last-child**

Select by Index

tag # ID-value **:nth-child(index)**

Select by Index

tag # ID-value **:nth-last-child(index)**

Pseudo Class – Type of child

First child of its type

tag # ID-value > tag # ID-value: **first-of-type**

Last child of its type

tag # ID-value > tag # ID-value: **last-of-type**

Nth child of its type

tag # ID-value > tag # ID-value : **nth-of-type**(index)

Pseudo Class

- `<tag>:checked` Selects checked `<tag>` elements (e.g. input)
- `<tag>:disabled` Selects disabled `<tag>` elements (e.g. button)
- `<tag>:enabled` Selects enabled elements (e.g. button)
- `<tag>:empty` Selects element that has no children
- `<tag>:hover` Selects `<tag>` elements on mouse hover
- `<tag>:only-of-type` Selects `<tag>` element that is the only `<tag>` of its parent
- `<tag>:only-child` Selects `<tag>` element that is the only child of its parent
- `<tag>::placeholder` Selects `<tag>` element with the place holder attribute specified
- `<tag>:invalid` Selects all `<tag>` with invalid value (e.g. input)
- `<tag>:valid` Selects all `<tag>` with valid value (e.g. input)
- `<tag>:link` Selects all unvisited links
- `<tag>:visited` Selects all visited links

[🔗 More pseudo classes](#)



/ **Mohammad Monfared**

Follow me for more like this