

XPATH

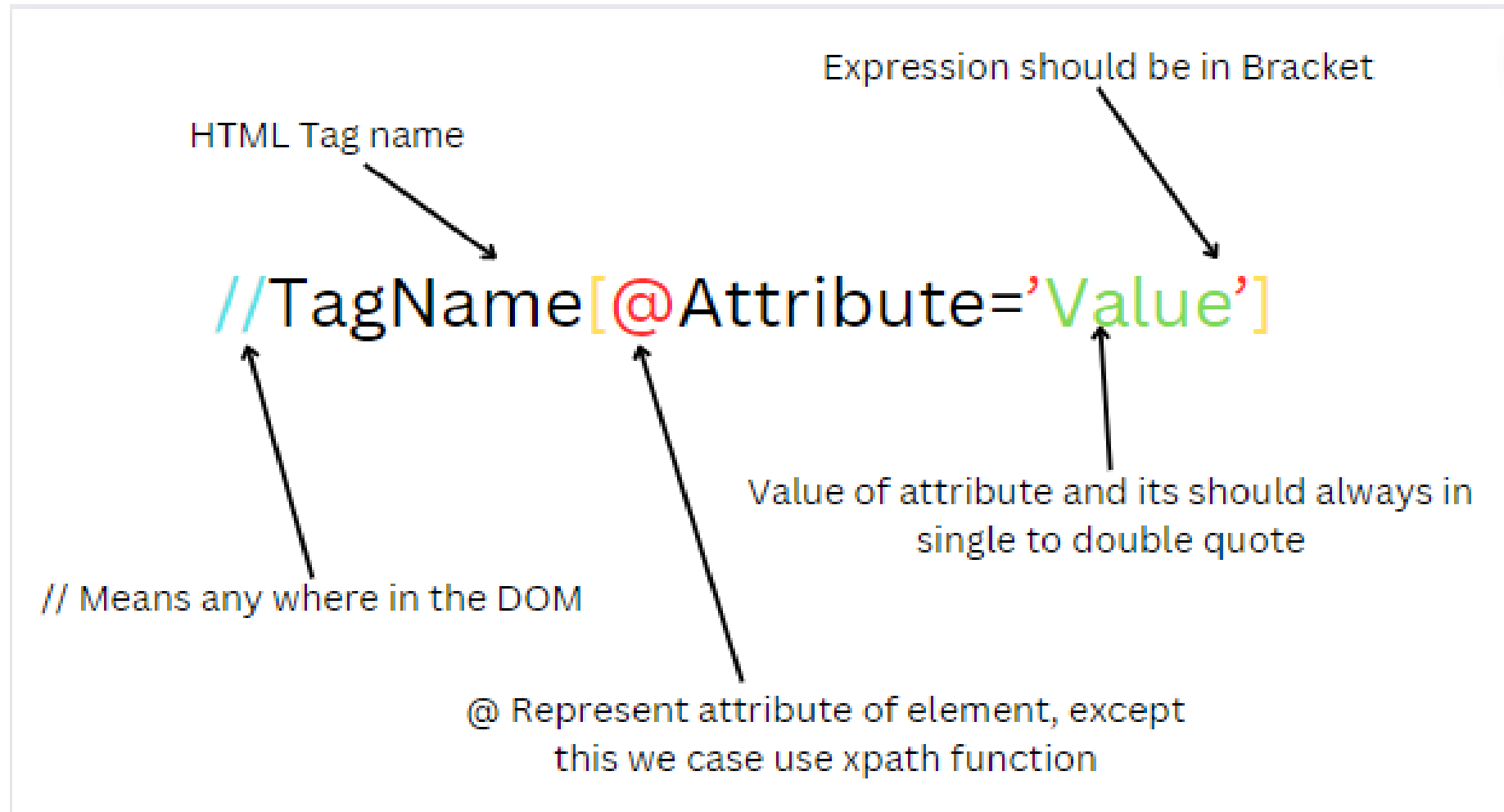
1. XPath Basics
2. XPath Functions
3. XPath Axes

XPath Basic

➤ XPath Syntax

- Selecting Nodes
- Selecting Unknown Nodes
- Selecting Several Paths
- Absolute and Relative
- Parents and Children
- Index
- Nested locators

XPath Syntax



Absolute and relative XPath

Absolute XPath

- The path from root element to targeted element without missing any elements in between.
- Absolute XPath starts from by using '/' (single forward slash) and '[index]' (square bracket with index).
- `/html/body/div[2]/div/div[1]/form/input[2]`

Relative XPath

- This XPath is specific to the target element. It uses expression to locate web element(s) in HTML documents.
- It uses "//" (double forward slash) and "/" along with other symbols, functions, and axes names.
- `//*[@id="file-submit"]`

Selecting Nodes

Index		
1	Node name	Selects all nodes with the name "node name"
2	/	Selects from the root node.
3	//	Select direct from given node
4	.	Selects the current node
5	..	Selects the parent of the current node
6	@	Selects attributes

Selecting Unknown Nodes

Index	Wild Card	Description
1	*	Matches any element node
2	@*	Matches attribute node

Selecting Several Paths

By using the | operator in an XPath expression we can select several paths.

Parents and Children

<code>//Tag[@condition]/..</code>	<code># Parent</code>
<code>//Tag[@condition]/Tag[condition]</code>	<code># Child</code>
<code>//Tag[@condition]/Tag[condition]</code>	<code># Search with in child)</code>

Index

<code>//Tag[@condition][index]</code>	<code>// nth child of its parent</code>
<code>(//Tag[@condition])[index]</code>	<code>// nth locator of this locator</code>

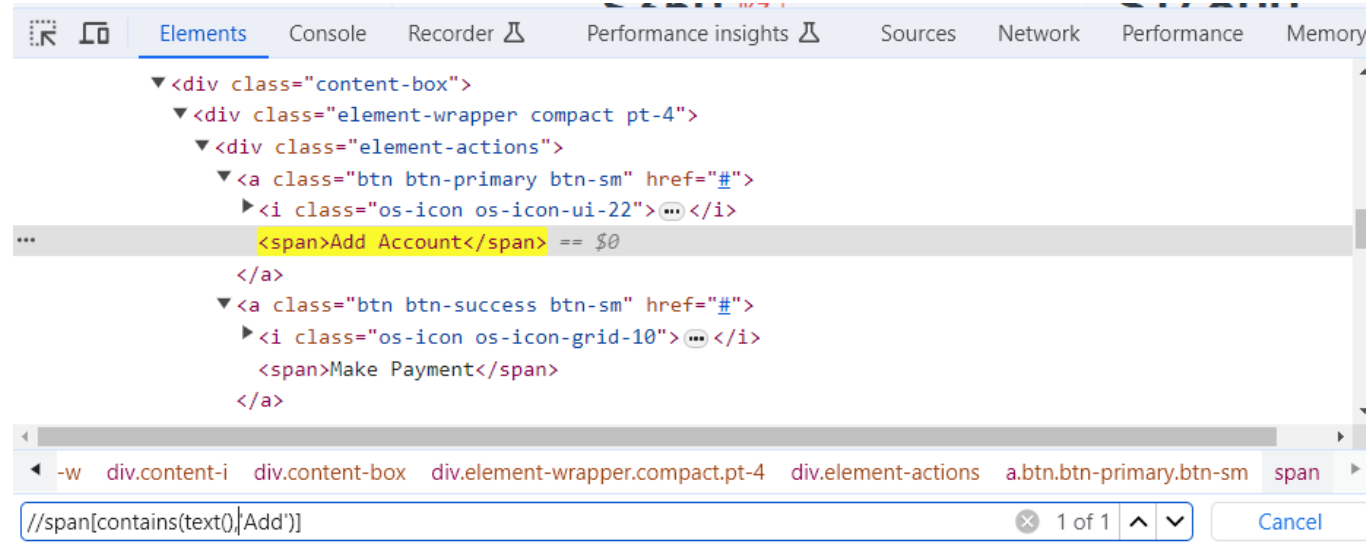
Nested locators

<code>//Tag[tag[@condition]]</code>	<code>// nested locator</code>
<code>//Tag[.//tag[@condition]]</code>	<code>// nested locator</code>

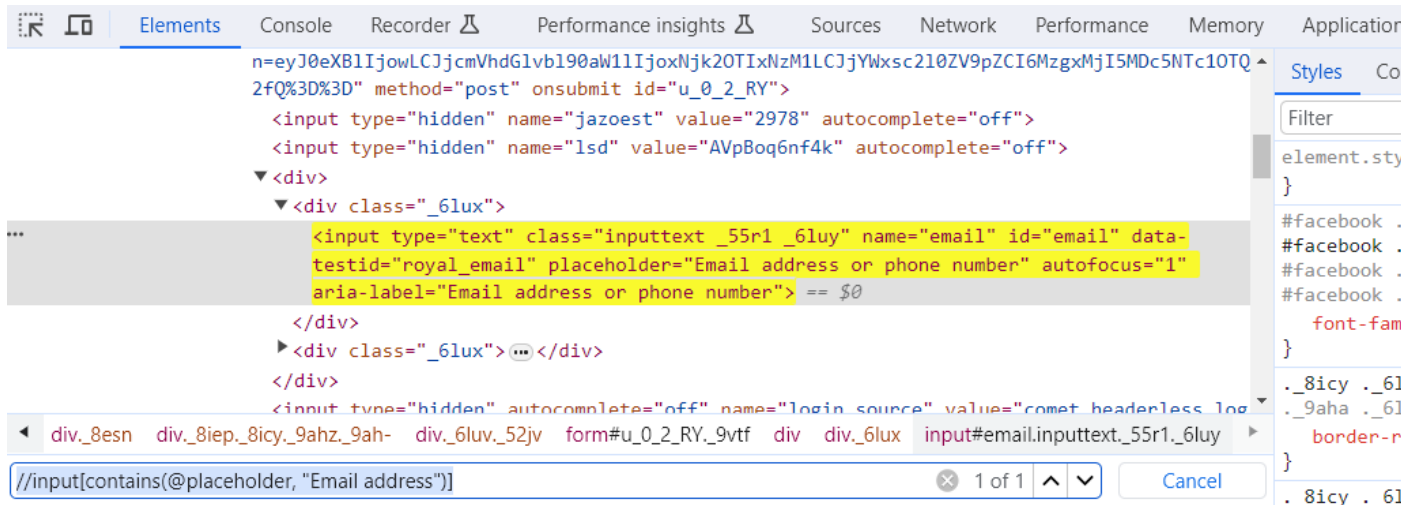
XPath Functions

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

contains()



`//span[contains(text(),'Add')]`



`//input[contains(@placeholder, "Email address")]`

starts-with()

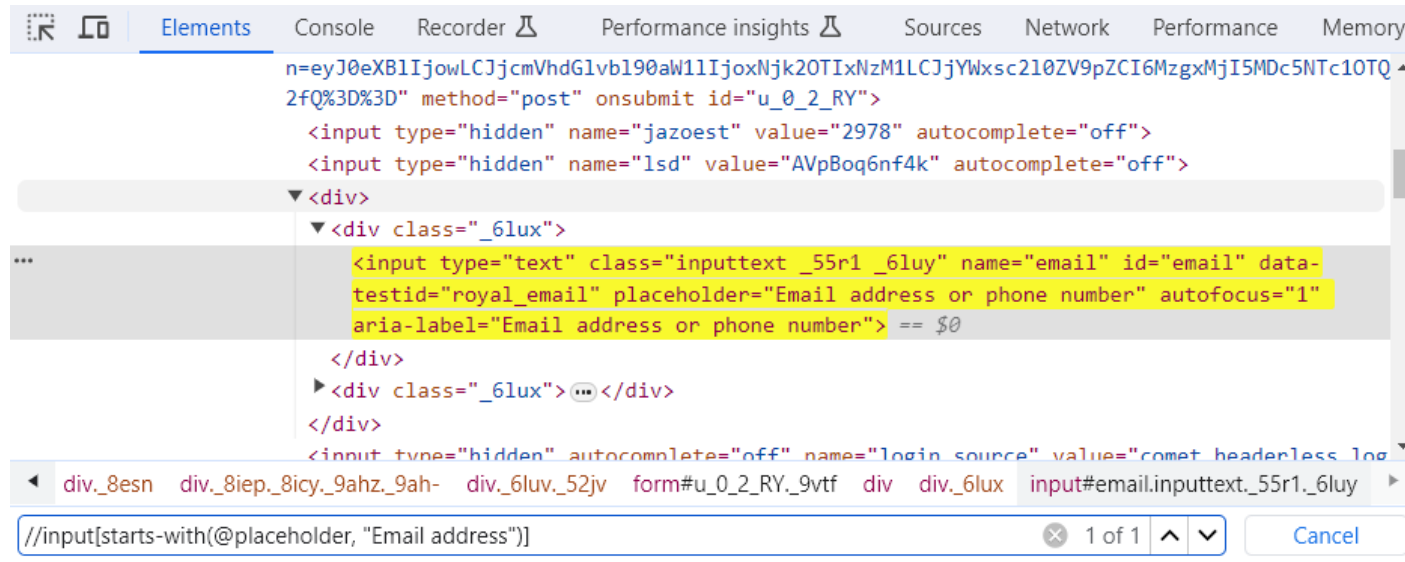


The screenshot shows the Chrome DevTools 'Elements' panel. The DOM tree is expanded to a `<a>` element. A search bar at the bottom contains the query `//span[starts-with(text(),'Add')]`. The search results show one match: a `` element with the text 'Add Account'.

```
...  
</a>  
<a class="btn btn-success btn-sm" href="#">  
  <i class="os-icon os-icon-grid-10">...</i>  
  <span>Add Account</span>  
</a>  
</div>  
<h6 id="time" style="text-align: center;font-size: x-large;color:red">Your nearest branch  
closes in: 30m 5s</h6>  
<h6 class="element-header">...</h6>  
<div class="element-box-tp">...</div>  
...
```

Search bar: `//span[starts-with(text(),'Add')]`

`//span[starts-with(text(),'Add')]`



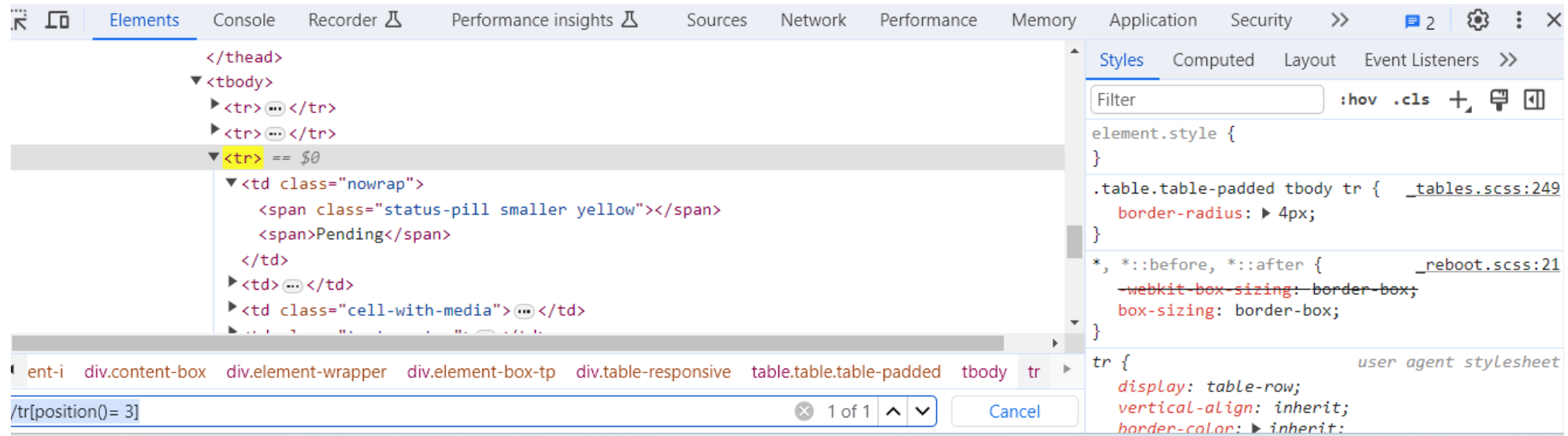
The screenshot shows the Chrome DevTools 'Elements' panel. The DOM tree is expanded to a `<div>` element. A search bar at the bottom contains the query `//input[starts-with(@placeholder, "Email address")]`. The search results show one match: an `<input type="text">` element with the placeholder text 'Email address or phone number'.

```
...  
<input type="hidden" name="jazoest" value="2978" autocomplete="off">  
<input type="hidden" name="lsd" value="AVpBoq6nf4k" autocomplete="off">  
<div>  
  <div class="_6lux">  
    <input type="text" class="inputtext_55r1_6luy" name="email" id="email" data-  
testid="royal_email" placeholder="Email address or phone number" autofocus="1"  
aria-label="Email address or phone number">  
  </div>  
  <div class="_6lux">...</div>  
</div>  
<input type="hidden" autocomplete="off" name="login_source" value="comet_headerless_log  
...
```

Search bar: `//input[starts-with(@placeholder, "Email address")]`

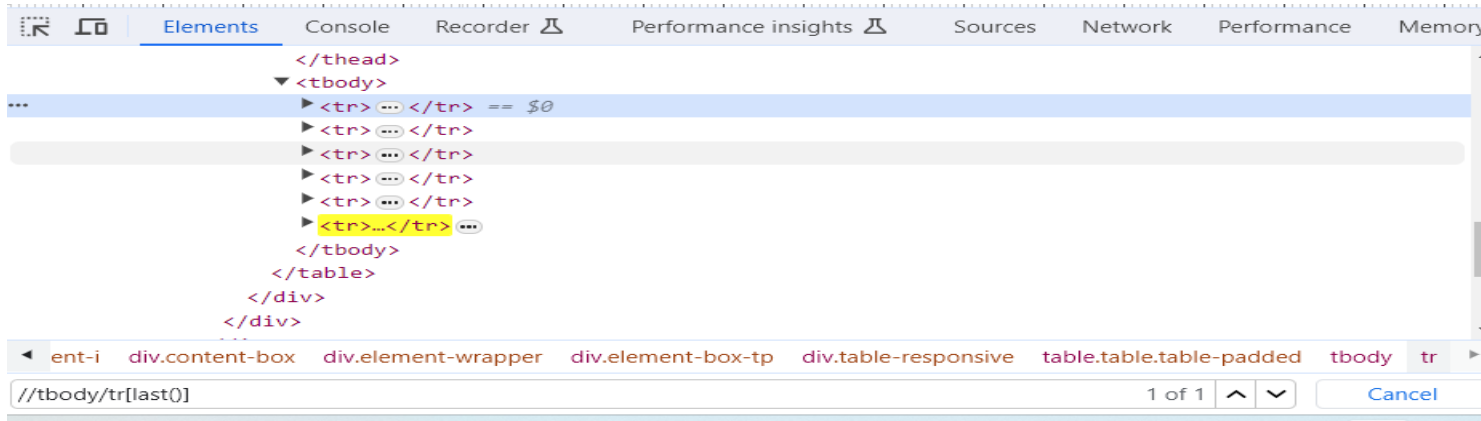
`//input[starts-with(@placeholder, "Email address")]`

position()



`//tr[position()= 3]`

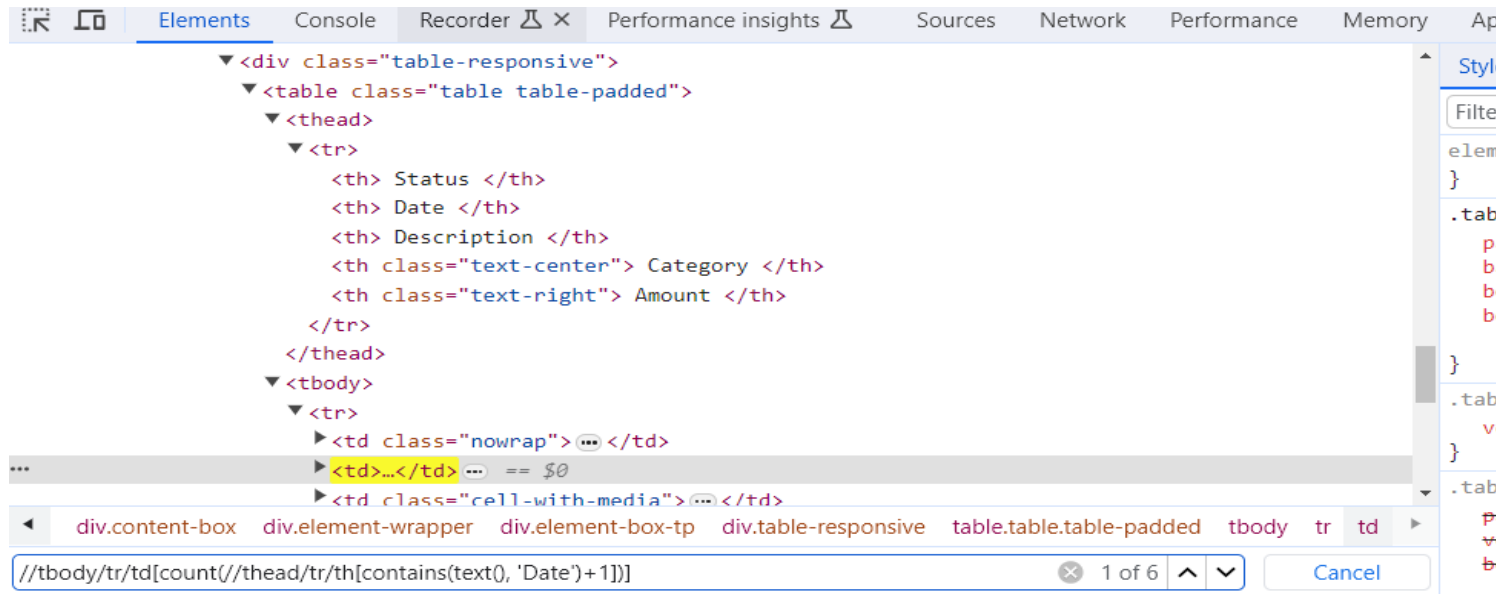
last()



//tbody/tr[last()]

count()

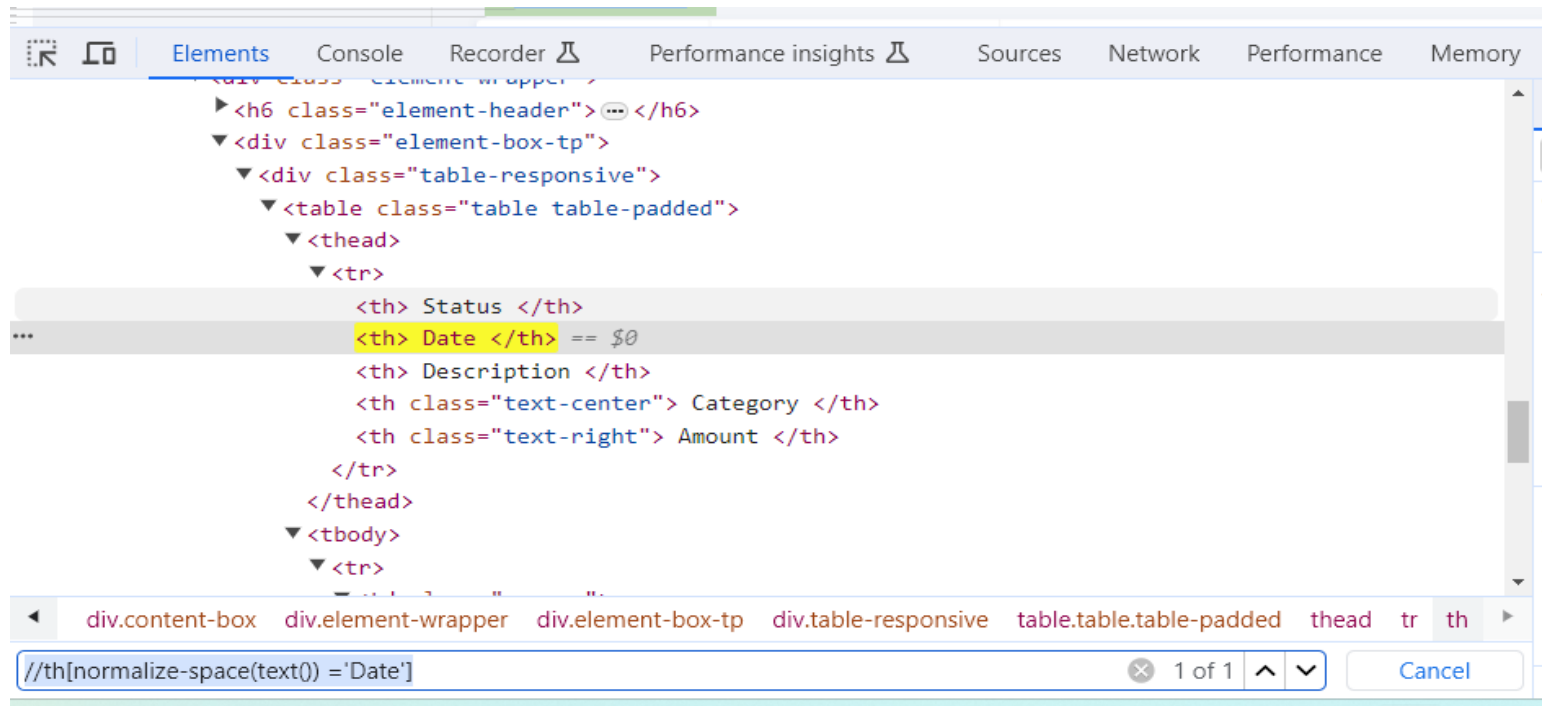
//tbody/tr/td[count(//thead/tr/th[contains(text(), 'Date')+1])]



normalize-space()

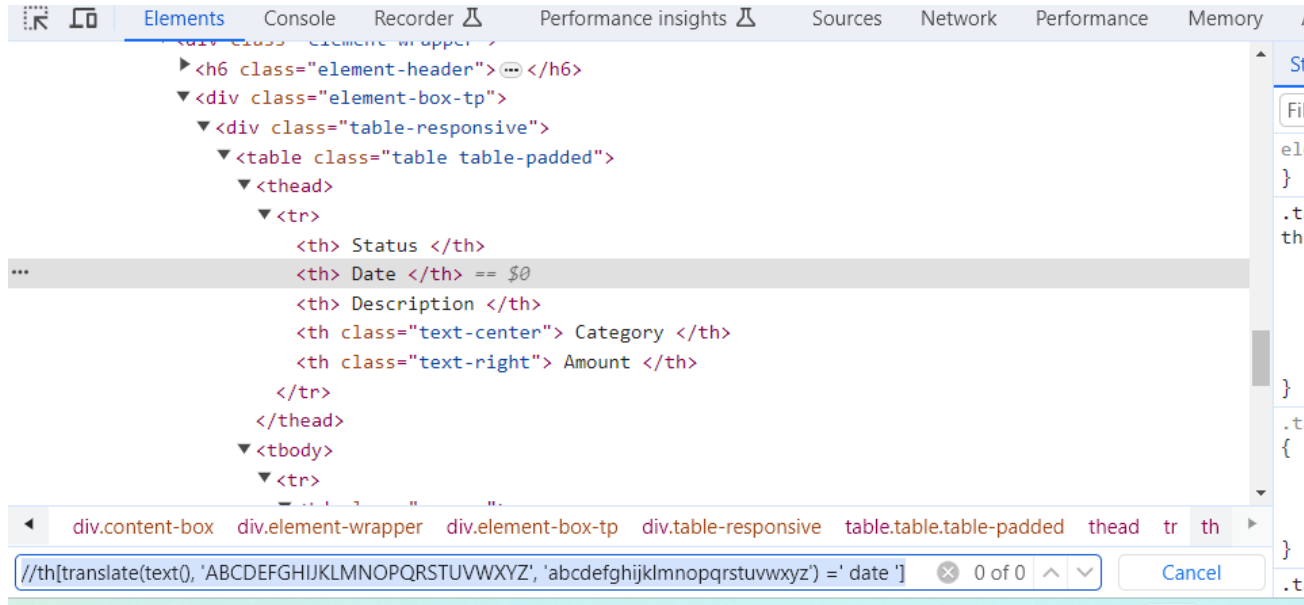
If an element has spaces in its text or in the value of any attribute, then to create an XPath for such an element we have to use the normalize-space function. It removes all the trailing and leading spaces from the string. It also removes every new tab or line existing within the string.

```
//th[normalize-space(text())='Date']
```



translate()

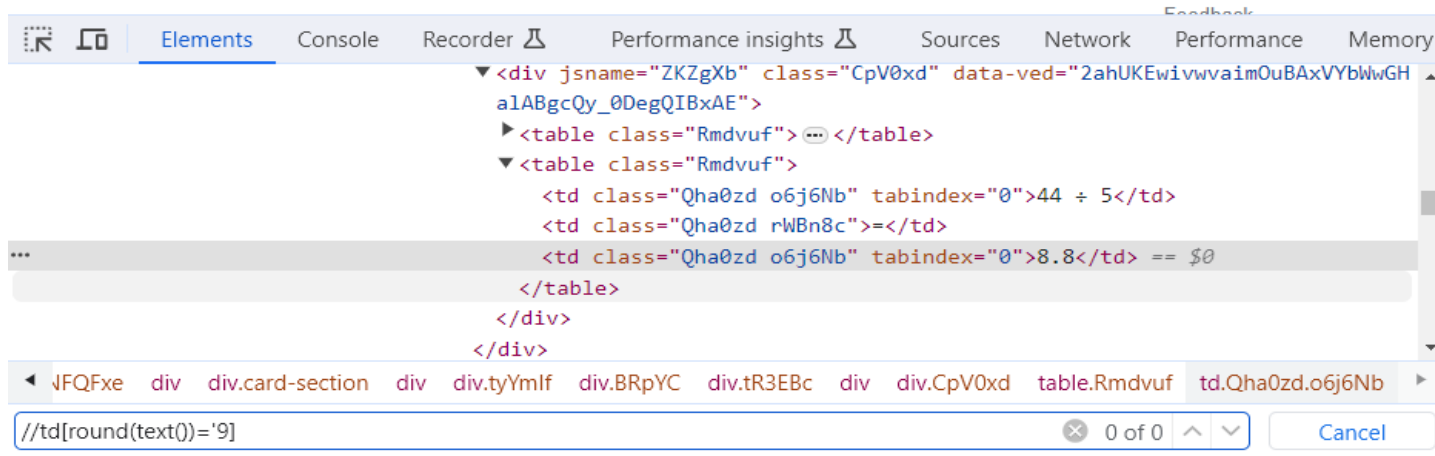
```
//th[translate(text(), 'ABCDEFGHIJKLMNOPQRSTUVWXYZ', 'abcdefghijklmnopqrstuvwxyz') =' date ']
```



```
//th[normalize-space(translate(text(), 'ABCDEFGHIJKLMNOPQRSTUVWXYZ', 'abcdefghijklmnopqrstuvwxyz')) ='date']
```

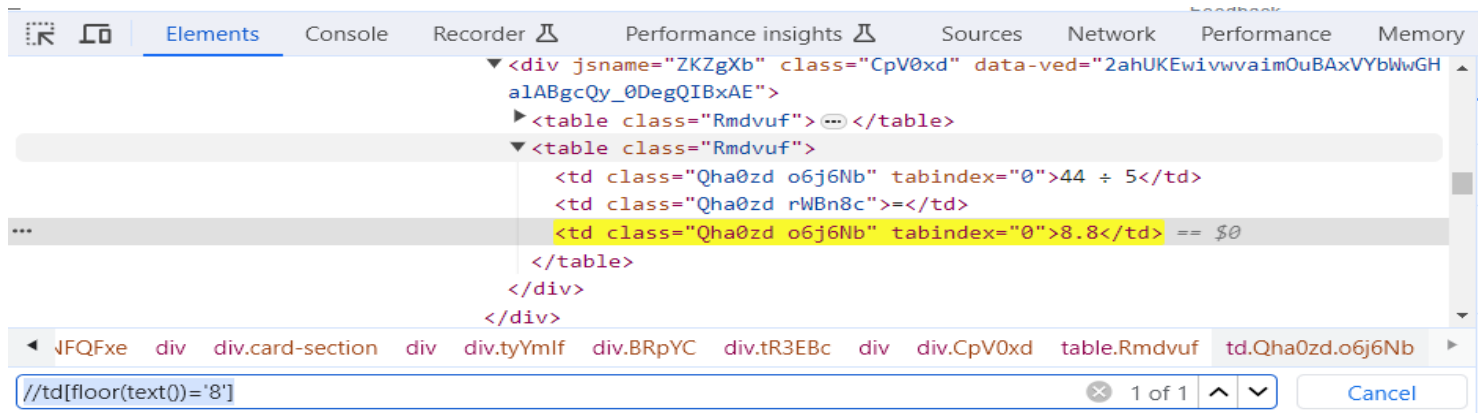
round()

```
//td[round(text())='9']
```



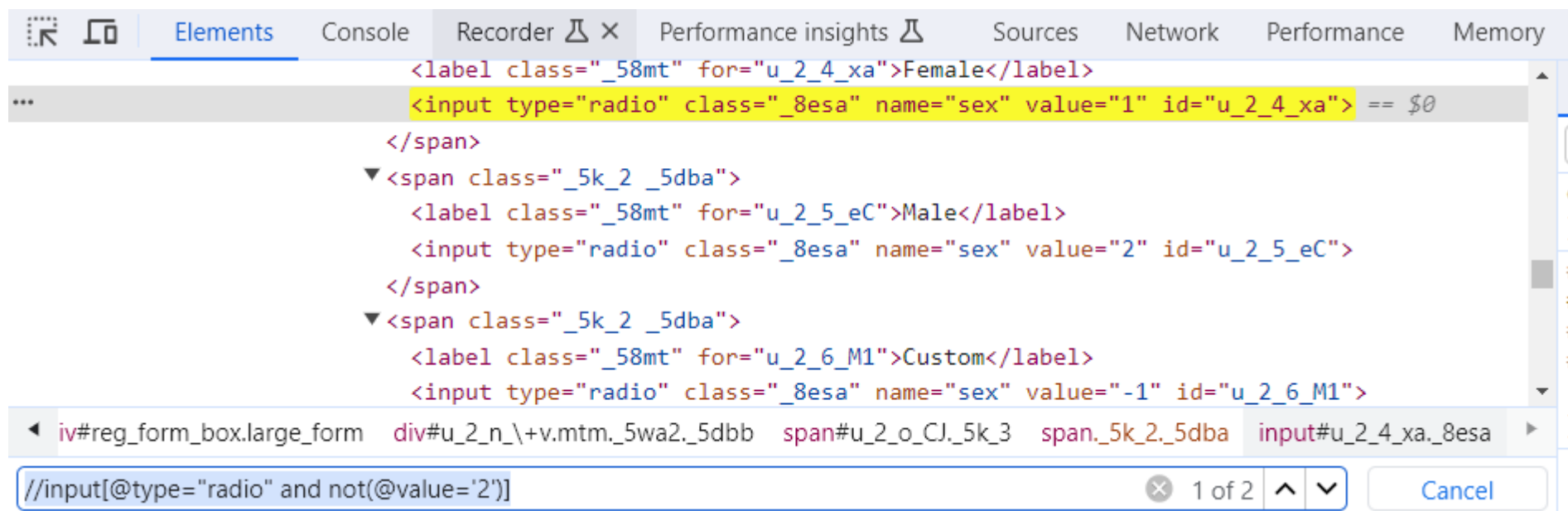
floor()

```
//td[floor(text())='8']
```



not()

`//input[@type="radio" and not(@value='2')]`

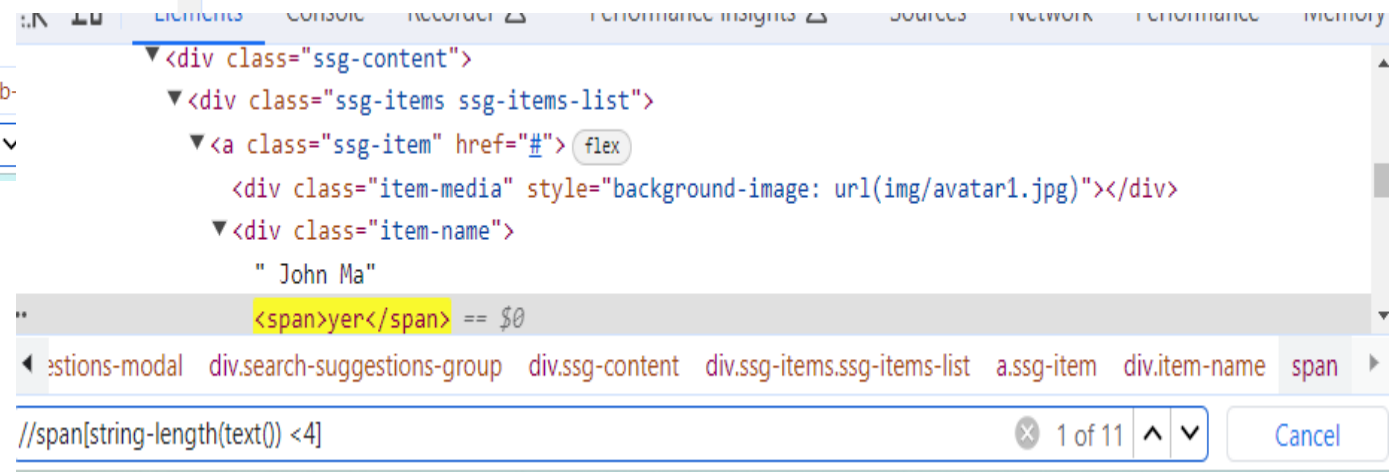
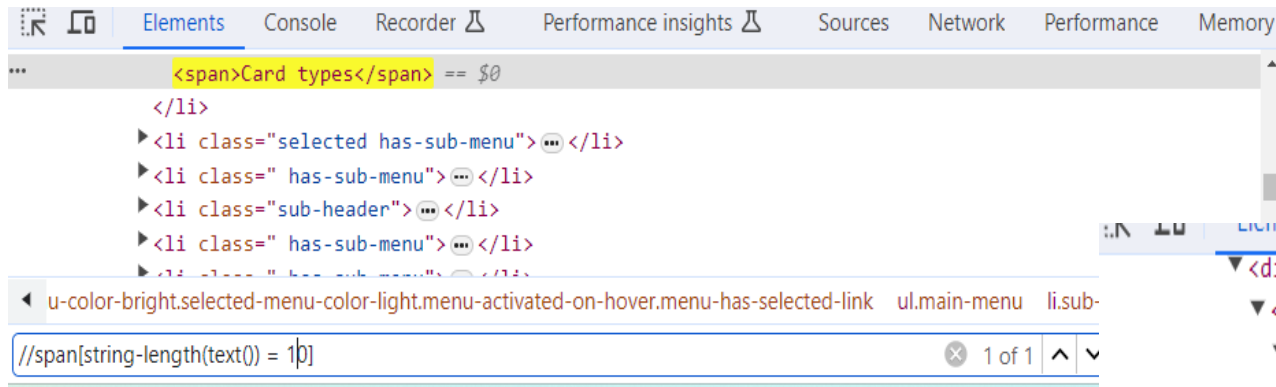
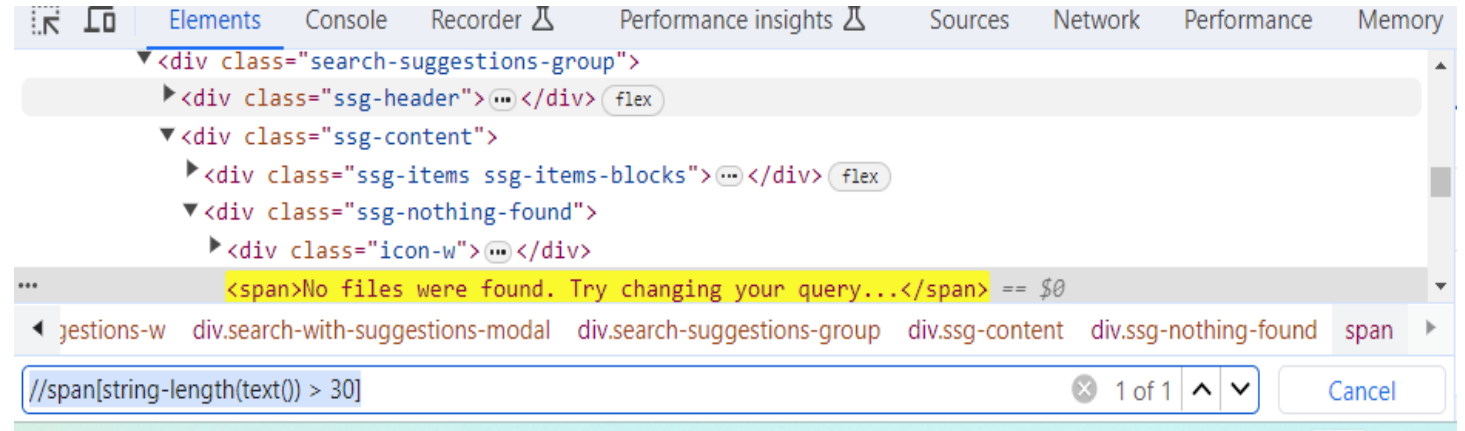


string-length()

//span[string-length(text()) = 10]

//span[string-length(text()) > 30]

//span[string-length(text()) < 4]

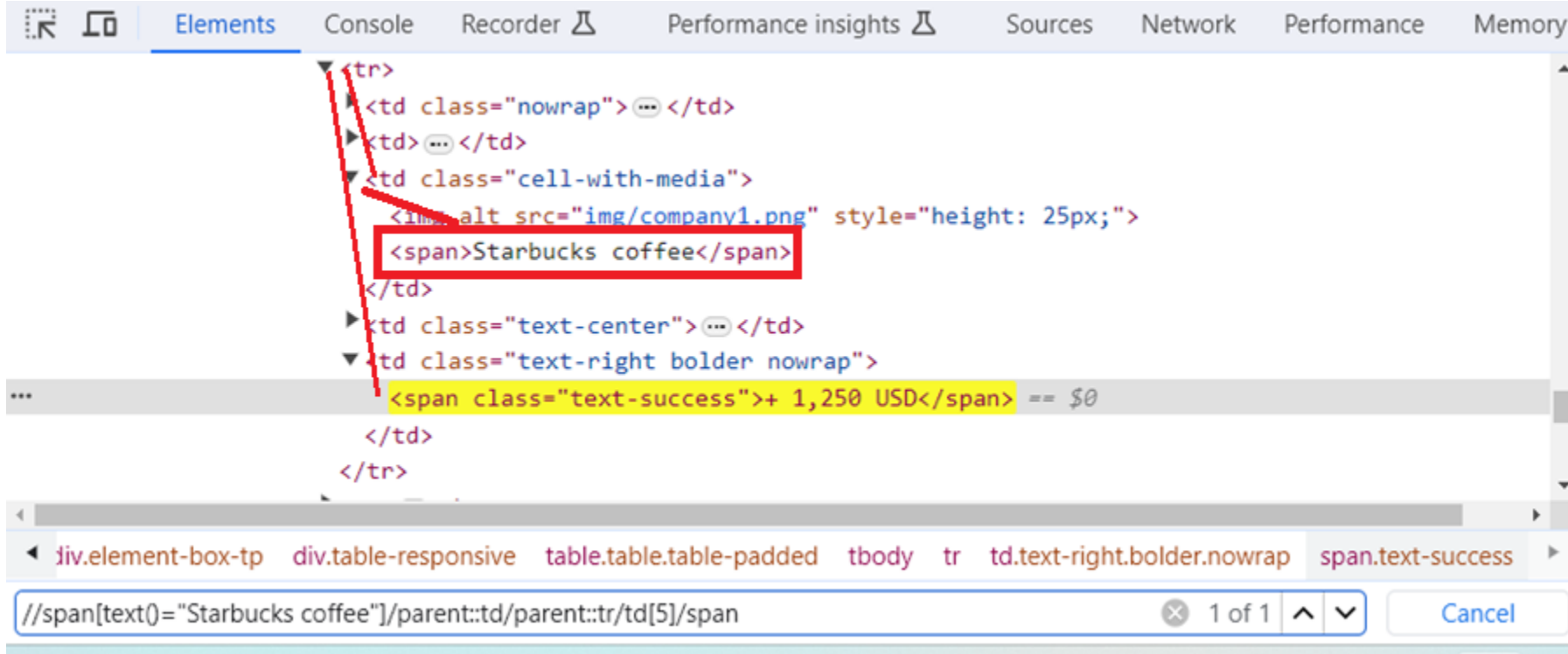


XPath Axes

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

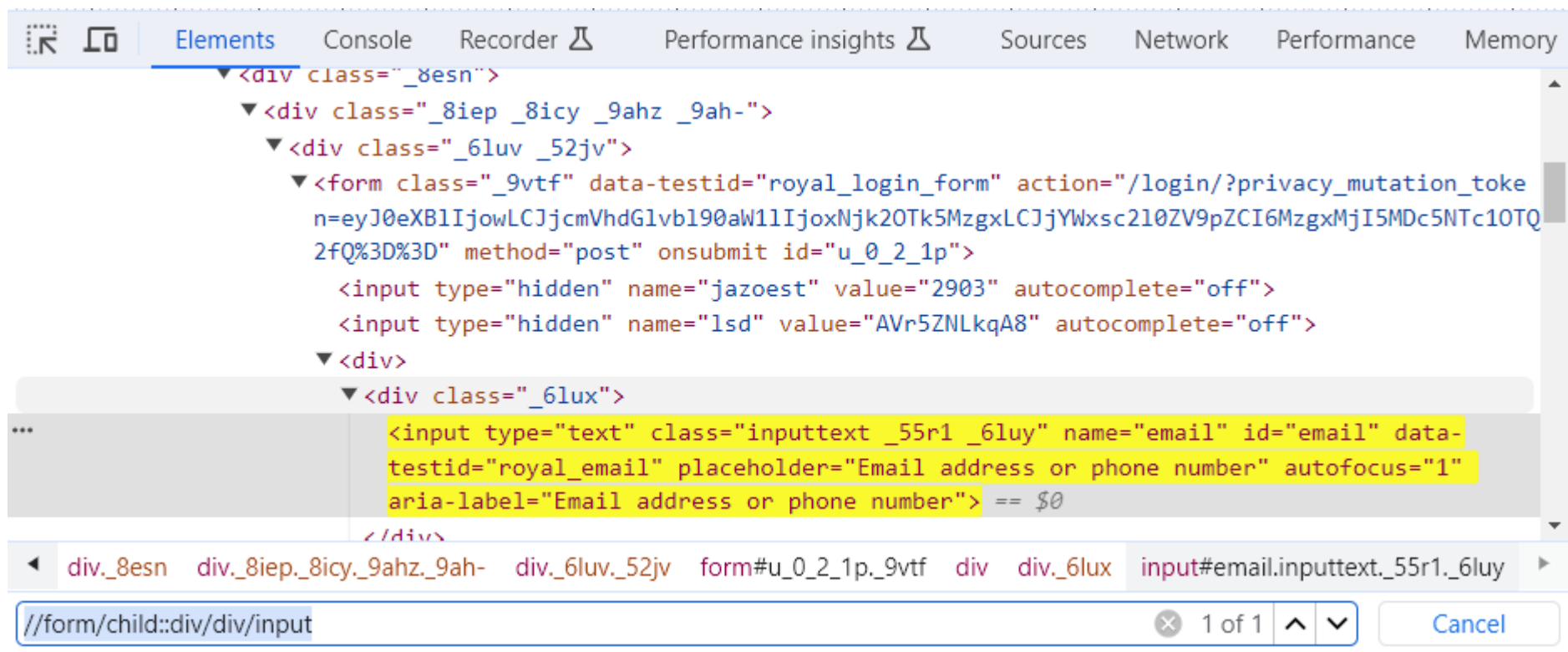
parent

`//span[text()='Starbucks coffee']/parent::td/parent::tr/td[5]/span`



child

//form/child::div/div/input

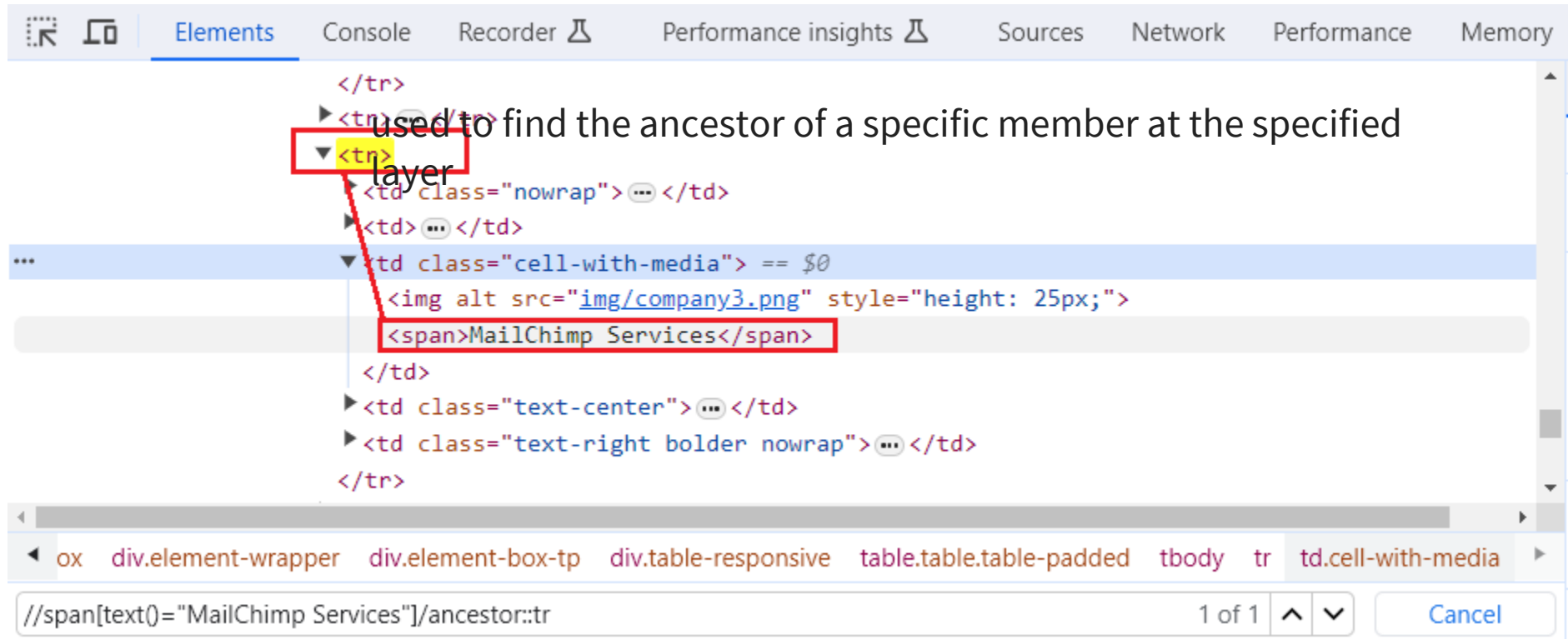


ancestor

used to find the ancestor of a specific member at the specified layer

```
//span[text()='MailChimp Services']/ancestor::table
```

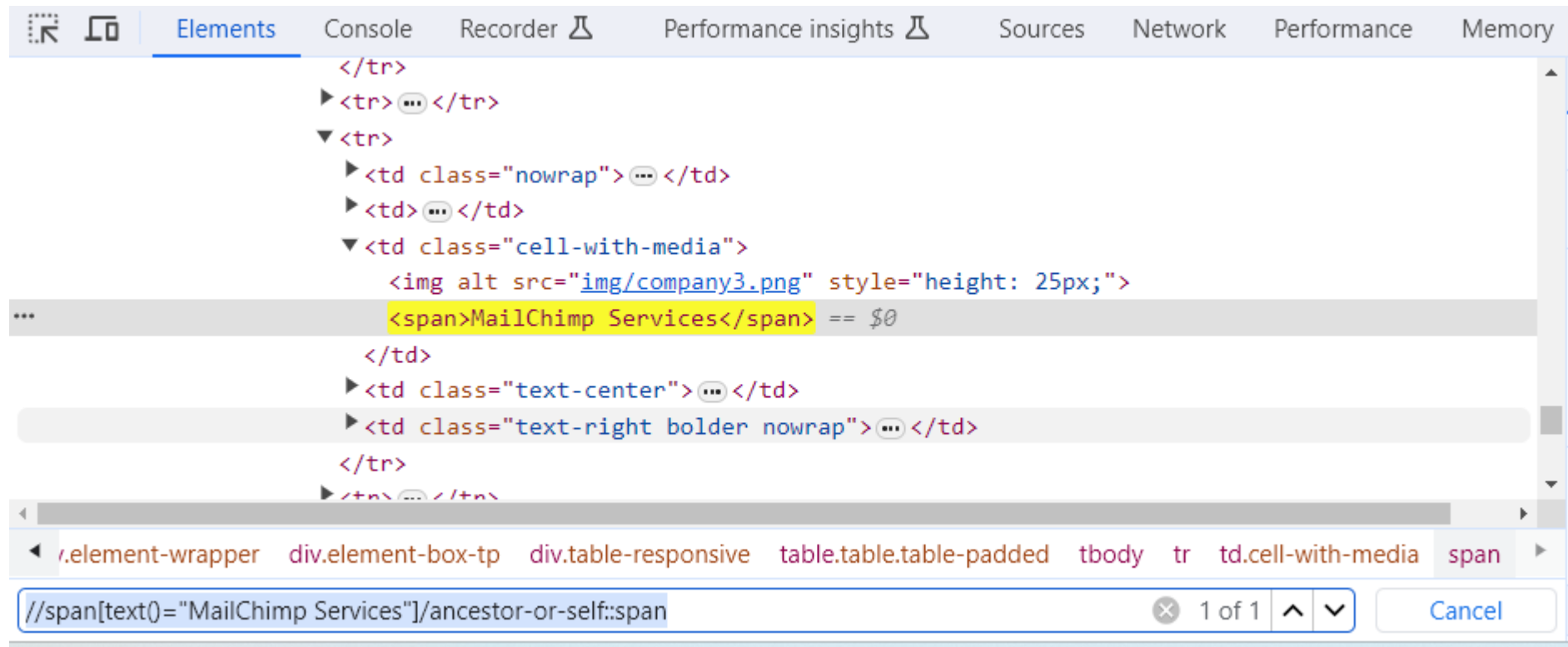
```
//span[text()='MailChimp Services']/ancestor::tr
```



ancestor-or-self

`//span[text()='MailChimp Services']/ancestor-or-self::span`

`//span[text()='MailChimp Services']/ancestor-or-self::td`

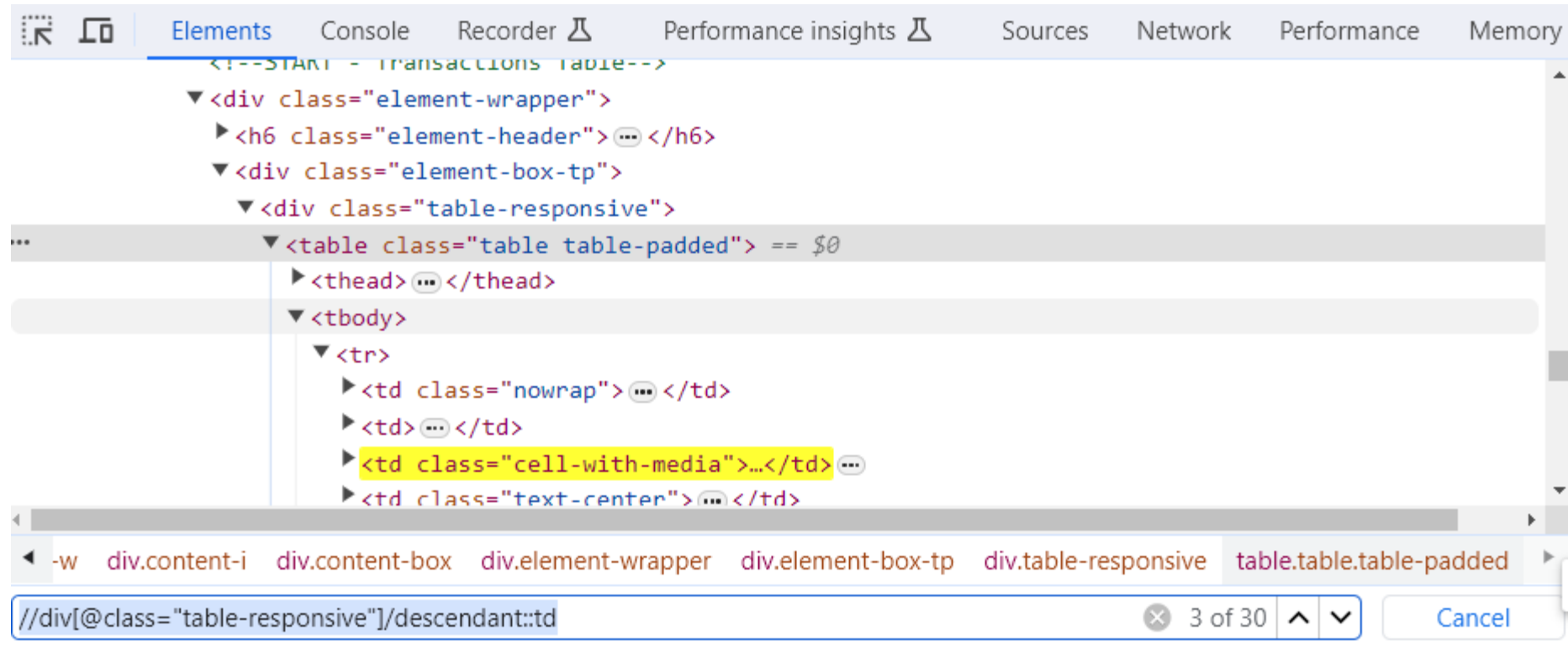


descendant

This function will return the descendant element of the particular element.

```
//div[@class="table-responsive"]/descendant::tr
```

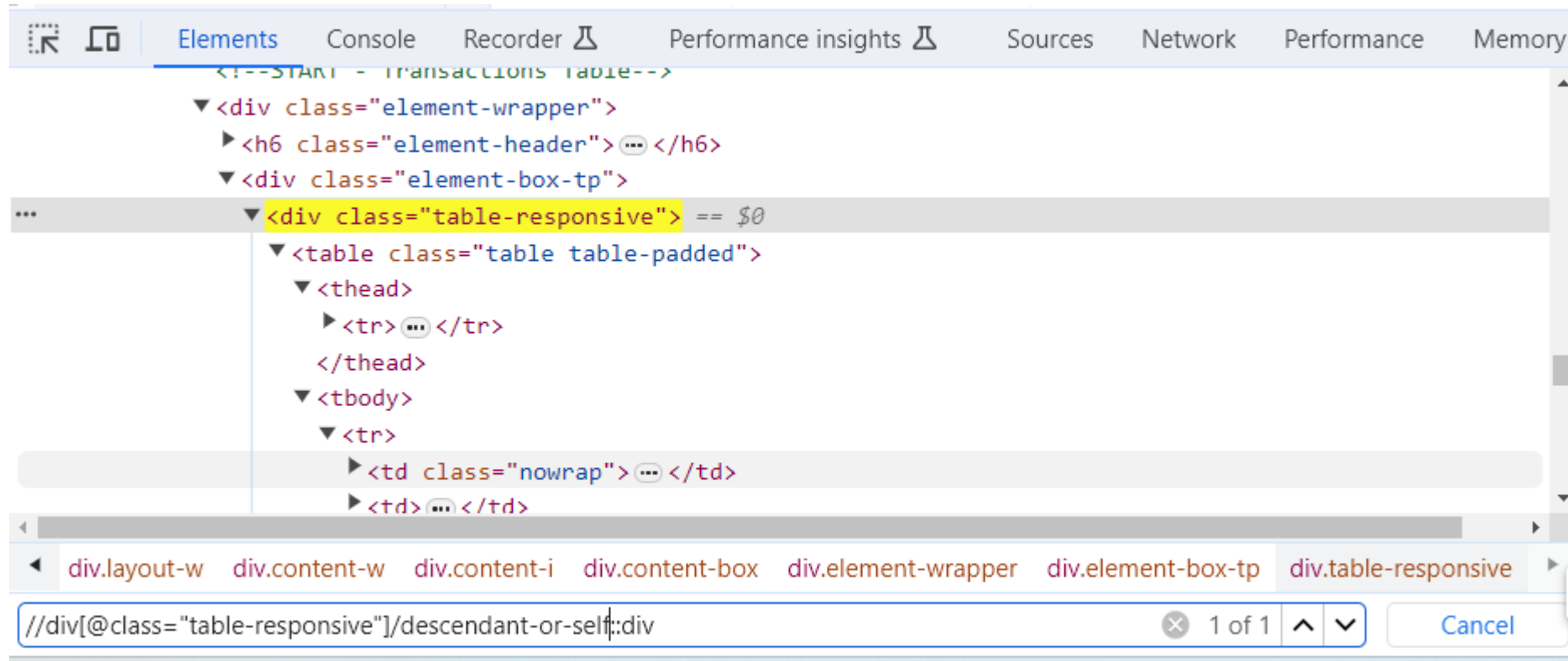
```
//div[@class="table-responsive"]/descendant::td
```



descendant-or-self

`//div[@class="table-responsive"]/descendant-or-self::div`

`//div[@class="table-responsive"]/descendant-or-self::td`

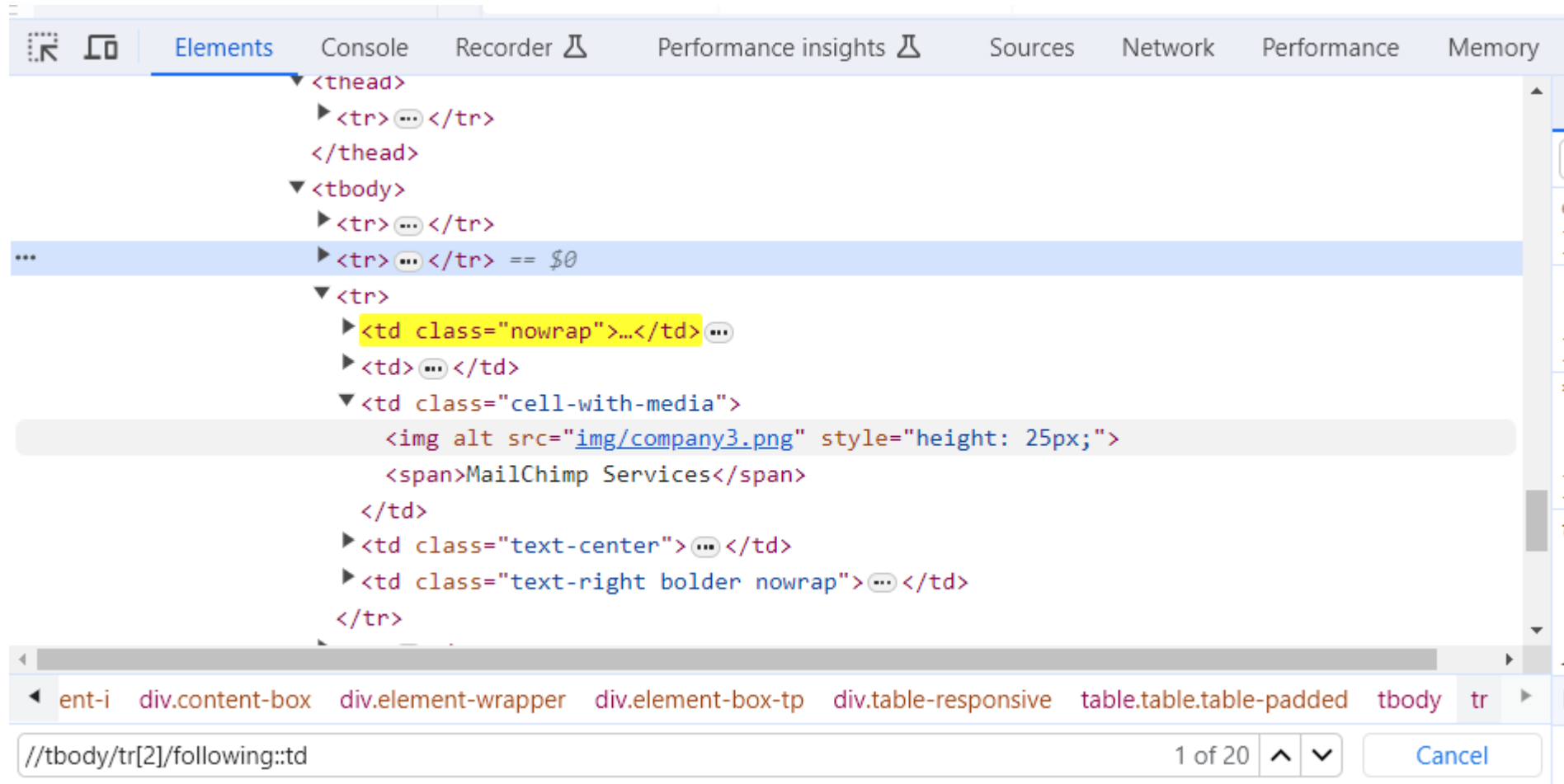


following

This function will return the immediate element of the particular component

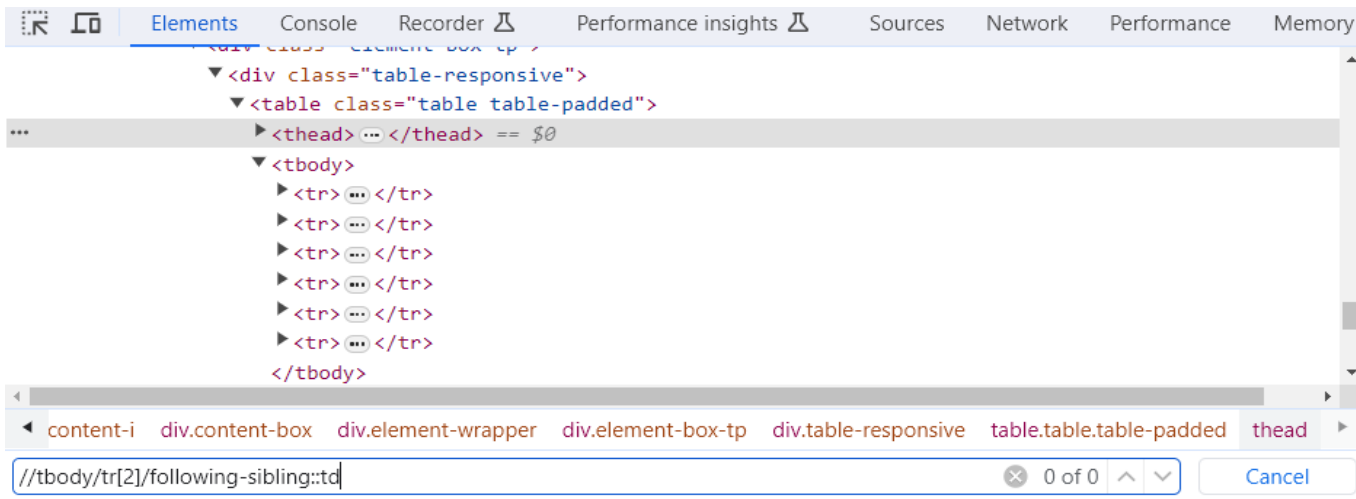
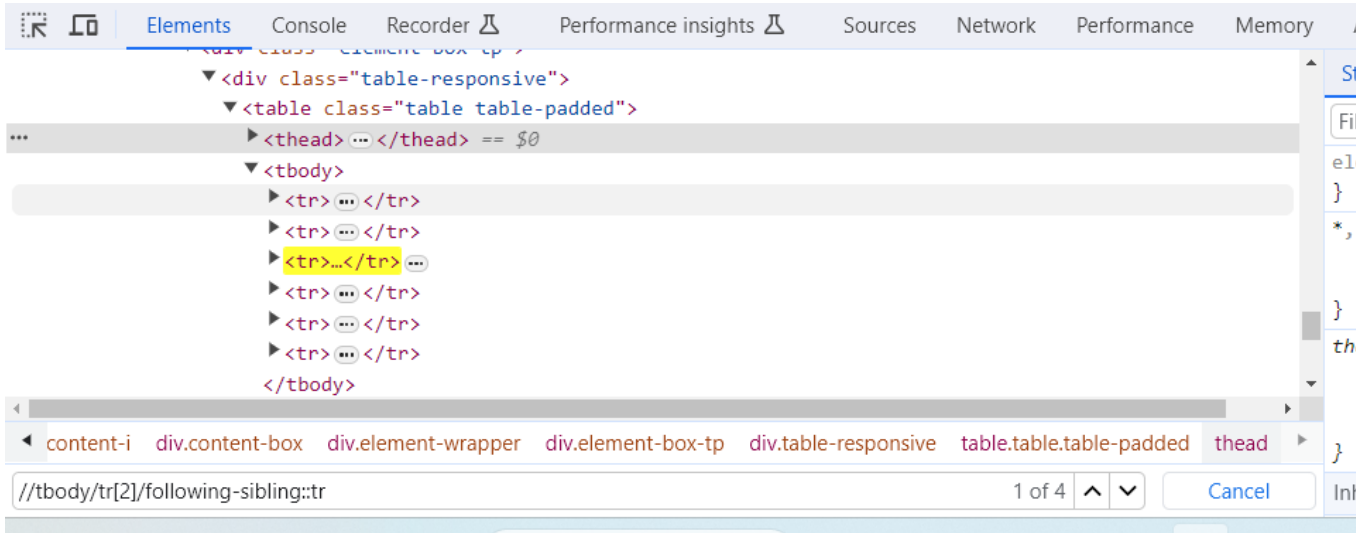
```
//tbody/tr[2]/following::tr
```

```
//tbody/tr[2]/following::td
```



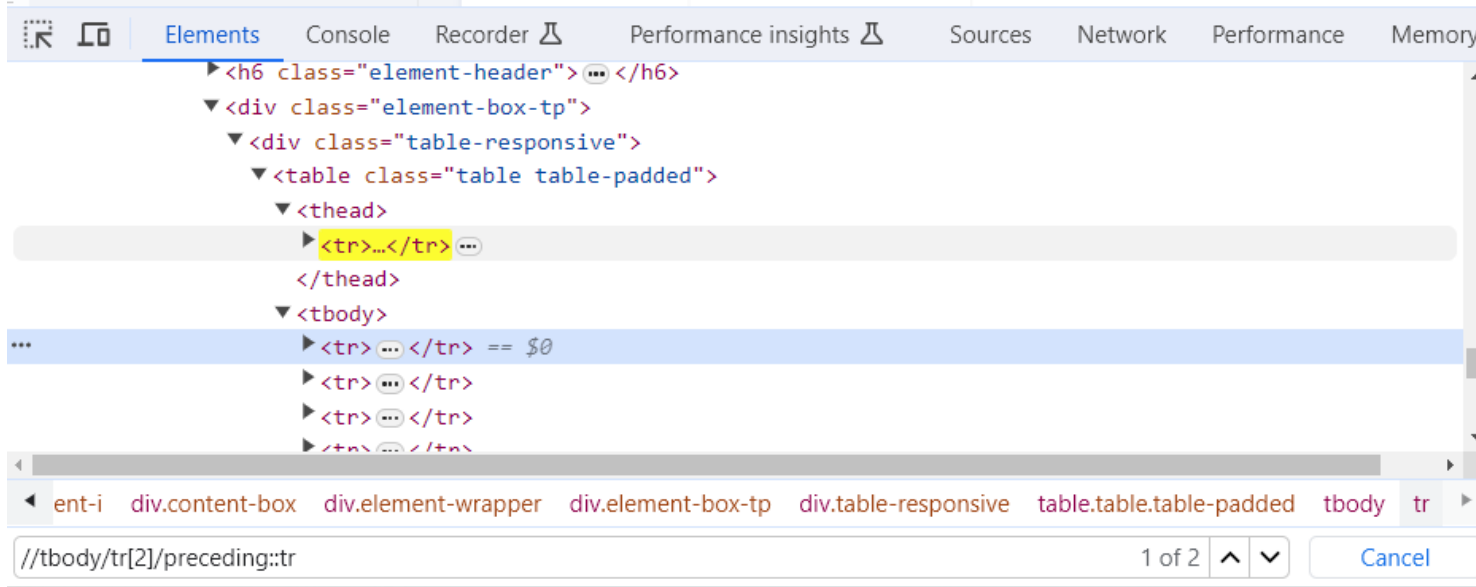
following-sibling

//tbody/tr[2]/following-sibling::tr

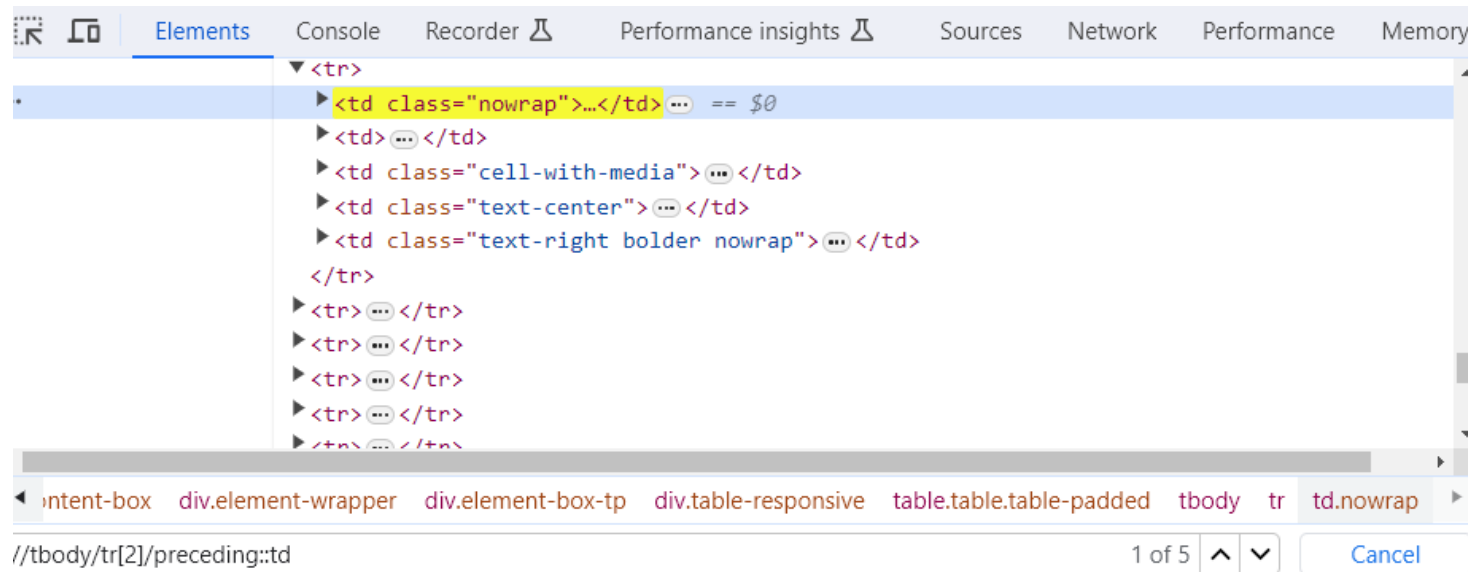


Preceding

This function will return the preceding element of the particular element



`//tbody/tr[2]/preceding::tr`



`//tbody/tr[2]/preceding::td`

THANKS FOR READING