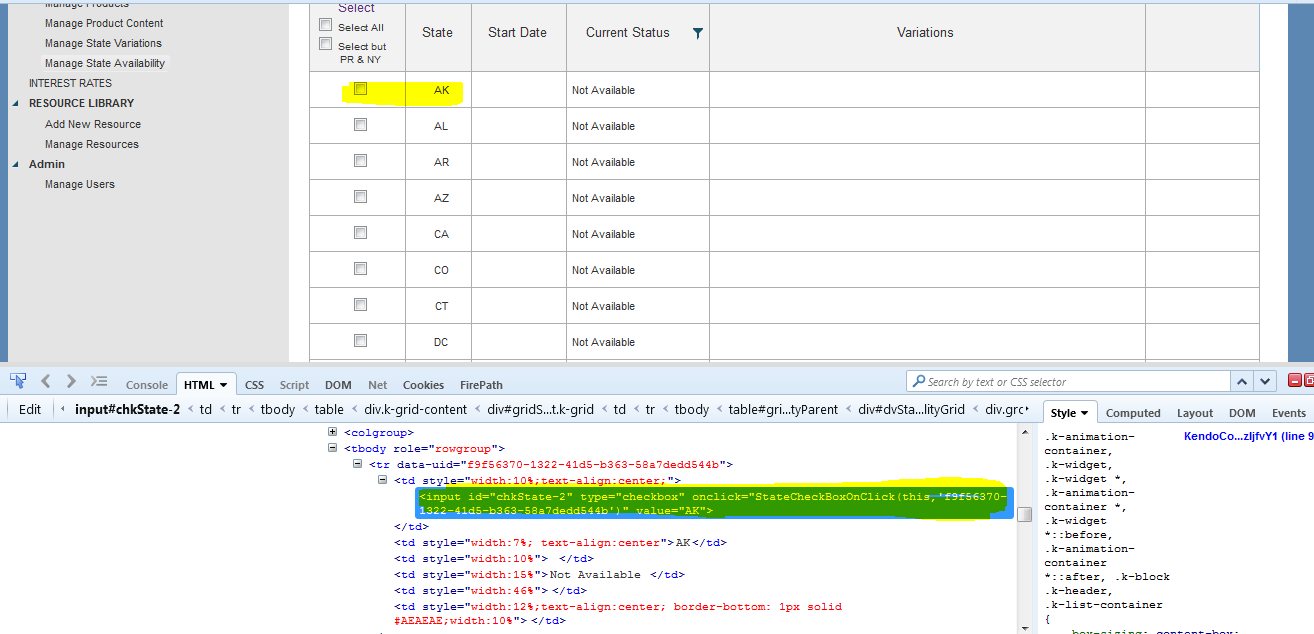
Xpath Examples:

1. How to use preceding-sibling

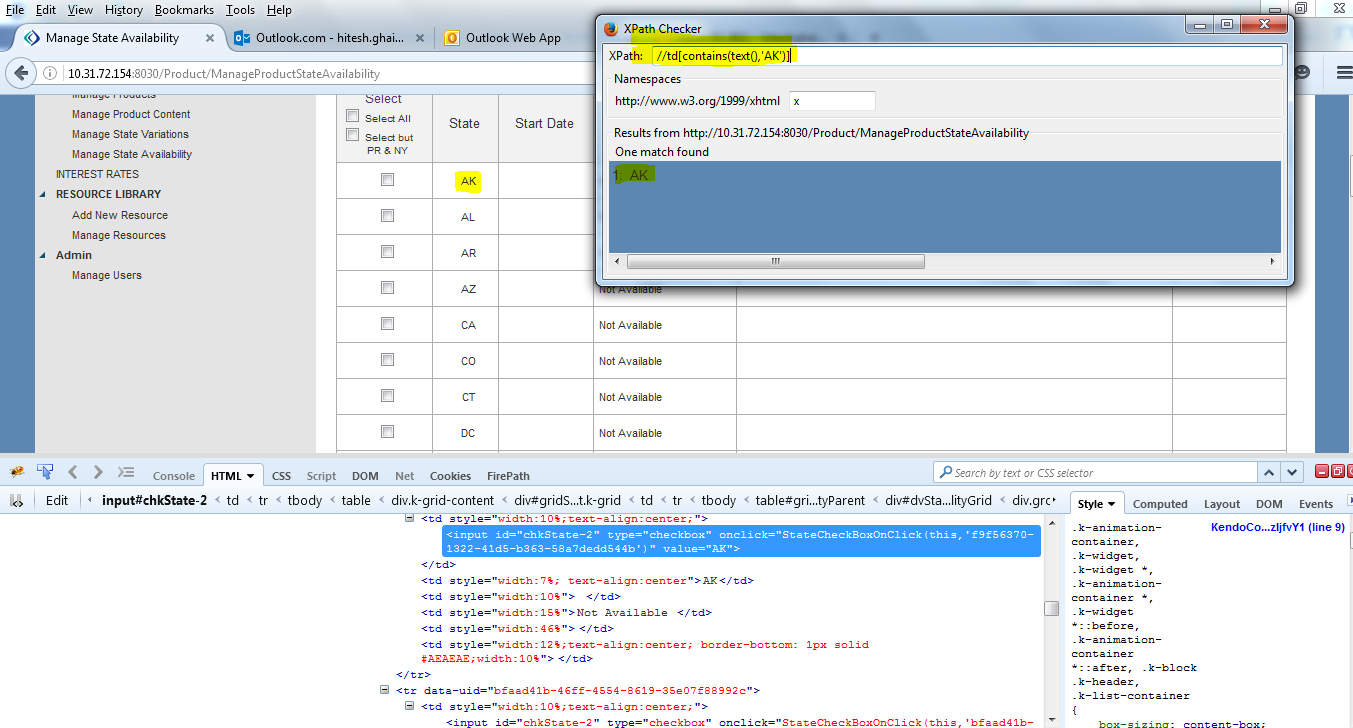
Preceding-sibling keyword select an element just before a element.

e.g In the following screen, user wants to get and then select a “AK” checkbox. It will be difficult without using the preceding or preceding-following keyword. So, this checkbox can be checked on the basis of its sibling which Text “AK” (Under State Col)



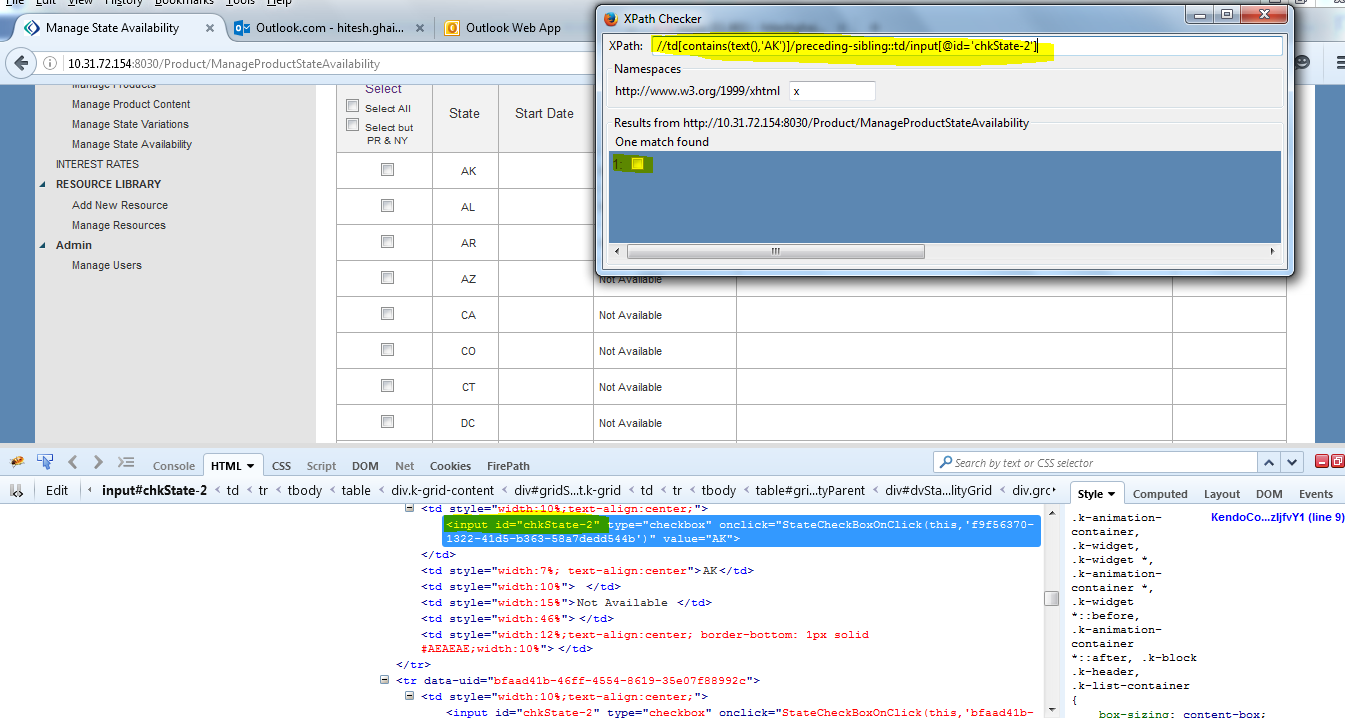
Step 1: First get ‘AK’ element (which is showing State Col) as follow:

//td[contains(text(),'AK')]

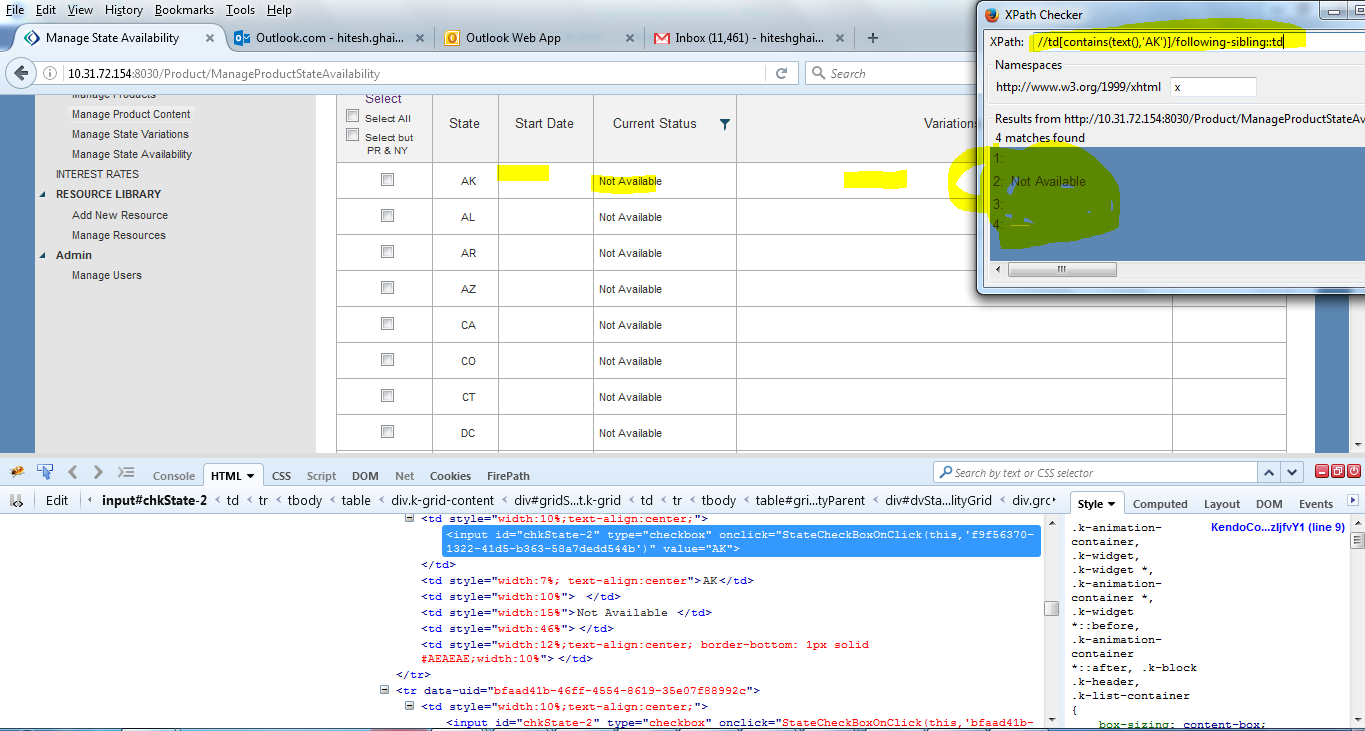


Step 2: Since ‘AK” checkbox is having “td” tag name and under which “input” its tag name. Now append “/preceding-sibling::td/input[@id='chkState-2']” to get “AK” checkbox

The full xpath will become “//td[contains(text(),'AK')]/preceding-sibling::td/input[@id='chkState-2']



**Example 2:** **“following-sibling”:** It selects the nodes or elements which is just comes after the elements or nodes. E.g if you want to select the elements after the “AK” (which is under state col) in the same row. Then “following-sibling” can be used to get all the td siblings in that row which are falling after “AK” text as follow:

’

1. Using parent

The below example will selects the parent node **of** the input tag **of** Id='email'. Ex: //input[@id='email']/parent::\*

//input[@id=’email’]/..

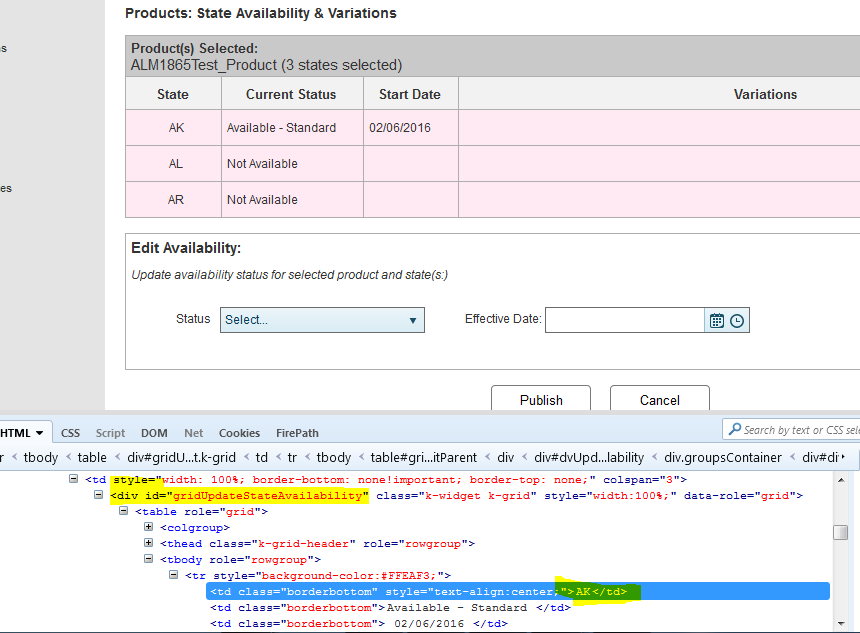
1. Following node :

It will identify the immediate node which start after the current node.

//input[@id='email']/following::\*

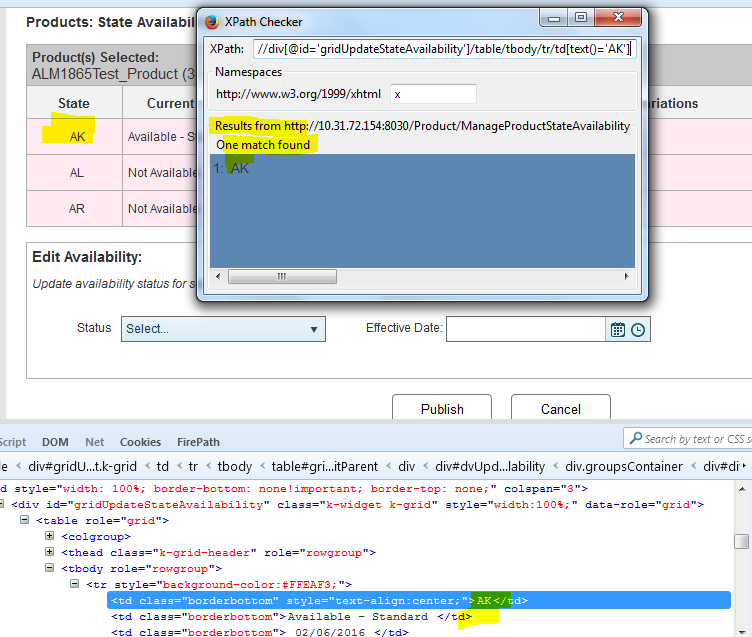
1. Usage of descendant: This ‘descendant’ keyword can be used to fetch element/node’s child. The main benefit of this keyword is to ignore the “between” childs and can go directly to required child and helps in shorten the xpath.

Look at the below example: Here, I want to get States names such as ‘AK’, ‘AL’ etc. These elements are falling under parent which is div @ id =’ gridUpdateStateAvailability’ as shown below:



So, the xpath can be like this: (which also working fine)

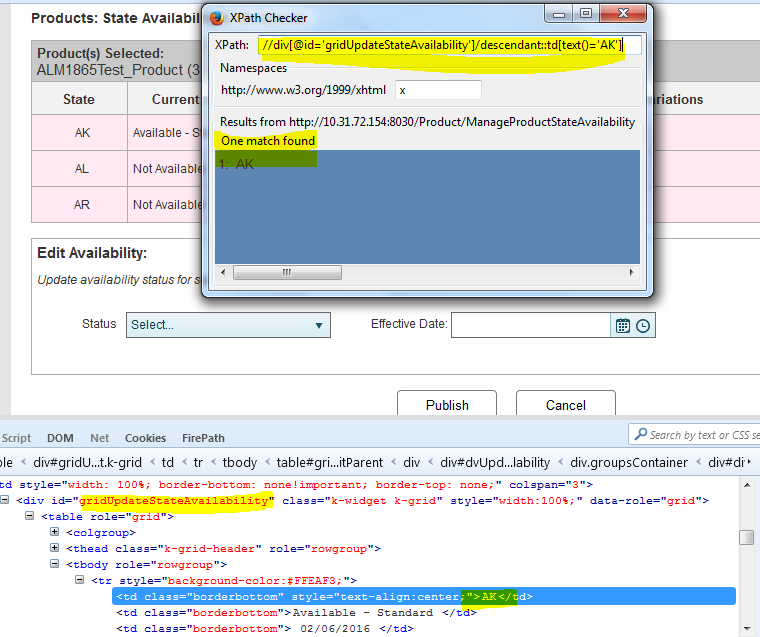
//div[@id='gridUpdateStateAvailability']/table/tbody/tr/td[text()='AK']



But above is long xpath where user has to navigate all the elements all through to reach the ‘td’ tag. So in order to shorten and prepare a rock solid xpath is to use ‘descendant’ keyword which works by ignoring the intermediate elements between the parent and required child element.

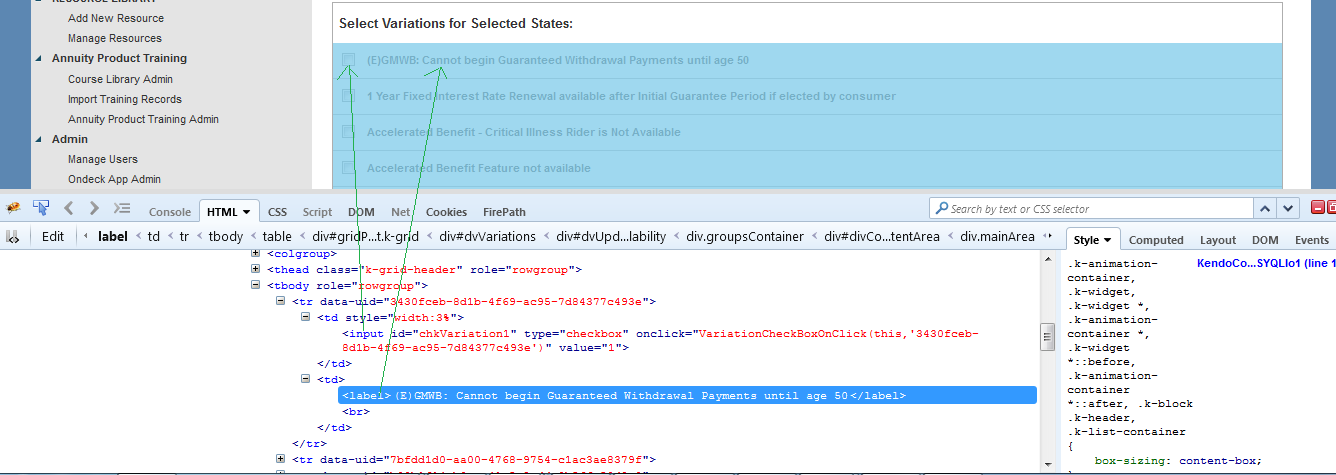
So, in order to avoid ‘table,’ tbody’ or tr elements between parent div and td, descendant keyword can be used as follow:

//div[@id='gridUpdateStateAvailability']/descendant::td[text()='AK']



Another Example of preceding-sibling

In the below example, we have to select a variation checkbox corresponding to various variations. Here, the checkbox tag name is “input” but we cannot get this checkbox element on the basis of input because it is not unique id.



So, the options left is to use preceding-sibling

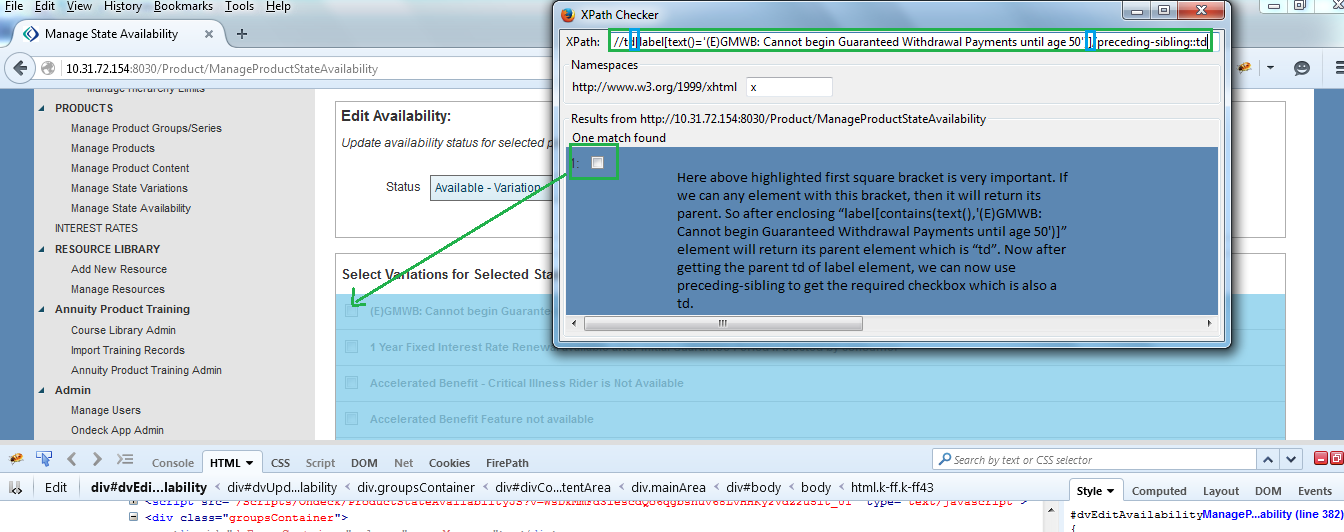
So, here xpath would be either of the following:

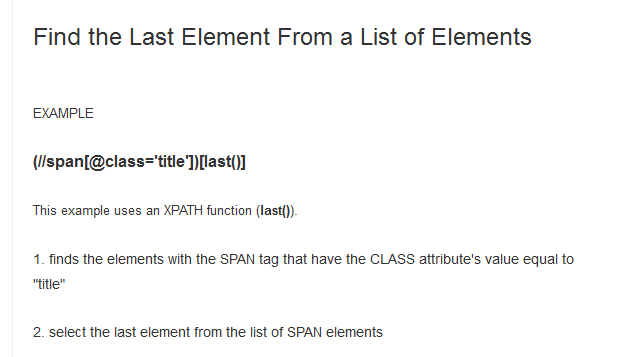
//td[label[text()='(E)GMWB: Cannot begin Guaranteed Withdrawal Payments until age 50']]/preceding-sibling::td

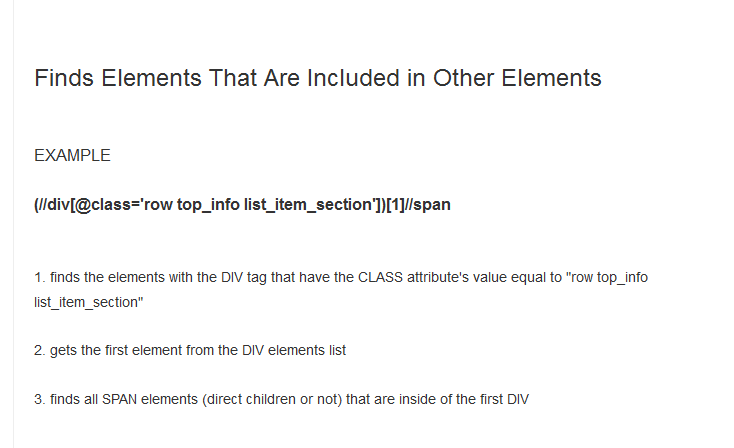
OR

//td[label[contains(text(),'(E)GMWB: Cannot begin Guaranteed Withdrawal Payments until age 50')]]/preceding-sibling::td

Here above highlighted first square bracket is very important. If we can any element with this bracket, then it will return its parent. So after enclosing “label[contains(text(),'(E)GMWB: Cannot begin Guaranteed Withdrawal Payments until age 50')]” element will return its parent element which is “td”. Now after getting the parent td of label element, we can now use preceding-sibling to get the required checkbox which is also a td.

s





\ 