

**Мануал по XPath**

[Введение](#h.jb66b5al5oaq)

[Необходимые инструменты](#h.lop4g36mu4wi)

[Как установить плагин в eclipse.](#h.clsq4oye8ref)

[Что такое XML](#h.7x8svilcjgw7)

[Что такое XPath](#h.wjea215ue21e)

[Использование XPath в разработке программ](#h.so4dxyayvfdl)

[Использование XPath в автоматизированном тестировании](#h.e556dtz2c3j9)

[Дерево документа](#h.je7j3ikja1gs)

[Узлы дерева](#h.9o8w1jc8ke17)

[Родственные связи](#h.a7apg5zhpv44)

[Предки и потомки](#h.z7cknq4pbarg)

[Родители и дочерние элементы](#h.fa3xyoj5epq2)

[Сестринские элементы](#h.f0mknqbu4fa9)

[Пути к элементам XPath](#h.w78otfwvsiuj)

[Абсолютный путь](#h.ehws9l2x0hd4)

[Относительный путь](#h.sptj1i8uagdp)

[Синтаксис языка XPath](#h.oxogf28u5uu7)

[Сокращенный синтаксис:](#h.igszg3j2hicq)

[Дочерний элемент (/)](#h.64zgvanf0ra9)

[Рекурсивный спуск (//)](#h.nsjhqfbzro2f)

[Фильтрация ([])](#h.g6gw7gfw8e26)

[Ссылка на все элементы (\*)](#h.7xpg34ghdsp)

[Атрибут (@)](#h.4wtoloe65j4z)

[Ссылка на родительский узел (..)](#h.jgvcet4j48o6)

[Группировка операций ()](#h.cqt5adoaploa)

[Не сокращенный синтаксис](#h.h2ztr9pozxjr)

[ancestor](#h.e31knzo9ev4y)

[ancestor-or-self](#h.sgdozwp9r8bi)

[attribute](#h.q0s53liqniak)

[child](#h.r8ufxnje22g2)

[descendant](#h.4yqyz5fj2fg6)

[descendant-or-self](#h.tsggo5jrlr2y)

[following](#h.a015y1dvlfw)

[following-sibling](#h.t6ksj2dvhx98)

[parent](#h.src9ax8fa9ix)

[preceding](#h.3ik8l16sebm5)

[preceding-sibling](#h.em33thusap7)

[self](#h.dwqc5jnxbhfv)

[Функции XPath](#h.ofqvq5tgt8u)

[Функции обработки наборов узлов](#h.yesiura1k7jy)

[node()](#h.eln4hpps2k68)

[text()](#h.cxtstco7f5x4)

[position()](#h.ity40ix6gqz3)

[last()](#h.sn4ygn1qvg13)

[count()](#h.aqygexpwdpjc)

[id()](#h.38dob9w4md0r)

[Строковые функции](#h.ptynk15942ms)

[string()](#h.joqenbrzp8ak)

[contains()](#h.gcd17kc5lyn)

[concat()](#h.o7va9mfv2hh5)

[string-length()](#h.95srhvbpxoz7)

[substring()](#h.emgzcprt3u0n)

[substring-before()](#h.asw4k81frnrf)

[substring-after()](#h.cto7alyeb03v)

[starts-with()](#h.9qc49q8n45ru)

[normalize-space()](#h.a4voypha83l2)

[translate()](#h.pazner1ml919)

[Логические функции](#h.e8pbqzvrol7a)

[логическое «или»](#h.b7yzd8j0uoyt)

[логическое «и»](#h.9gk3h9kjkeia)

[логическое «равно»](#h.8xo39490xhr)

[логическое «меньше»](#h.19khfekf20ud)

[логическое «больше»](#h.6bsp6chkw759)

[логическое «больше либо равно»](#h.uc0dph7w9zgu)

[Булевы Функции](#h.9amhls988snw)

[boolean()](#h.pzv5pevdmlyz)

[true()](#h.wxid7os934g9)

[false()](#h.izkh4boa3szh)

[not()](#h.xj46pnyzt0vs)

[Числовые функции](#h.hcgaxjha3u77)

[Cложение](#h.x9rnp7l689qd)

[Вычитание](#h.b2yvgixd743j)

[Умножение](#h.dzhjsg79cps6)

[div](#h.if3p1d5ksdi4)

[mod](#h.gtmt5jgt9y00)

[number()](#h.7z2a14honz8q)

[sum()](#h.olh51lssnnkc)

[floor()](#h.4maoz56tcn03)

[ceiling()](#h.6tmnqmq6fkuj)

[round()](#h.sauovdo47kbp)

[Советы по написанию хорошего XPath](#h.9hivnhj7p8jm)

[1. Постарайтесь избегать абсолютных путей в XPath запросах.](#h.ucail9fwgzth)

[2. Старайтесь не использовать цифры.](#h.a8kni2iucmi9)

[3. Не использовать длинные XPath.](#h.gyc3vpic4xmh)

[4. Cтарайтесь не использовать \* перед квадратными скобками.](#h.p0iccg45ko2i)

[5. Cтарайтесь не использовать подзапросы.](#h.qc7szb9o67w)

[Примеры сложных XPath](#h.qkrwckq52qpd)

[Полезные ссылки](#h.150c1foh3jug)

[Ответы на задания](#h.5ynw9jwjvpw)

# Введение

Данный мануал рассматривает основные аспекты XPath (XML Path Language). Он предназначен для людей, которые никогда не сталкивались с XPath. В данном документе материал изложен в понятном формате, что облегчит изучение XPath начинающему пользователю.

По окончанию изучения документа, вы сможете составлять сложные XPath.

# Необходимые инструменты

# 

Для всех упражнений данного мануала мы будем использовать HTML страничку, расположенную по адресу [www.xml2selenium.com/xpath/](http://www.xml2selenium.com/xpath/)

Перед изучением данного мануала, вам потребуется установить:

1. браузеры: Chrome или Firefox

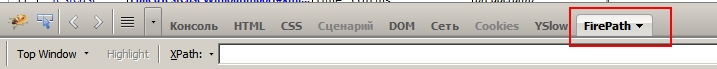
Если вы будете пользоваться браузером Chrome, то пункт 2 пропустите

2. для браузера Firefox вам потребуется установить расширения [firebug](https://addons.mozilla.org/ru/firefox/addon/firebug/) и [firepath](https://addons.mozilla.org/ru/firefox/addon/firepath/)

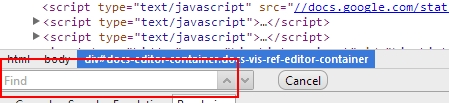
после установки данных расширений, возле адресной сроки появиться иконка firebug

1.jpg

при нажатии на нее откроется окно firebug. В дальнейшем, мы будем работать с расширением firePath



3. что бы открыть инструменты разработчика в Chrome, нажмите сочетание клавиш ctrl+shif+I. После этого, в нижней части экрана откроется панель. Что бы открыть строку для ввода XPath, нажмите на любое место в панели и нажмите клавиши ctrl+f



Примечание: все примеры будут приводиться в fireBug в Firefox, но вы можете использовать любой браузер, принцип работы одинаковый.

Так же во время изучения материала, вы можете пользоваться плагином для eclipse. Ссылки на плагин:

Для версий Eclipse (Galileo, Helios, Indigo, Juno, Kepler) - <http://eclipse-xpath-evaluation-plugin.googlecode.com/svn/trunk/eclipse-xpath-evaluation-plugin-update-site/>

Для версии Eclipse (Ganymede) - <http://eclipse-xpath-evaluation-plugin.googlecode.com/svn/trunk/eclipse-xpath-evaluation-plugin-update-site/>

# 

# [Как установить плагин в eclipse](https://code.google.com/p/eclipse-xpath-evaluation-plugin/wiki/Installation).

# Что такое XML

XML — это язык разметки, похожий на HTML. XML был создан для описания

данных. Теги XML не предопределены. Вы можете использовать (создавать) свои теги. ***Тег -*** это слово, заключённое в треугольные скобки <>, тег должен открываться (<пример>) и закрываться (</пример>), так же бывают одиночные теги (<пример/>).

Пример:

<full\_name>

<first>Petr</first>

<last>Petrov</last>

</full\_name>

/>

XML не был создан для выполнения каких-либо действий.

Важно понимать, что XML не выполняет никаких действий, а просто

содержит в себе некие данные. Этот язык разметки был создан для структурирования, хранения и передачи информации. Следующий пример является адресом Иванова Ивана Ивановича, который проживает на материке Евразия, в части света Европа, в стране Беларусь, в городе Солигорск, по ул. Ленина д. 17 подъезд 5 в кв. 20 представленной в XML:

<?xml version="**1.0**" encoding="**UTF-8**"?>

<continent>

<part\_of\_world value="**Evrazia**">

<countrie value="**Belarus**">

<sity value="**soligorsk**">

<adress>

<street value="**Lenina**"/>

<home>**17**</home>

<front\_door>**5**</front\_door>

<room value="**20**"/>

<man id="**Ivanov**">

**Ivanov Ivan Ivanovich**

</man>

</adress>

</sity>

</countre>

</part\_of\_world>

</continent>

Как видите, язык XML очень лаконичен.

Континент (**<**continent**>**) содержит в себе часть света(<part\_of\_world>) с параметром в котором содержится название (value="**Evrazia**") части света, в части света содержиться страна (<countrie>) с параметром в котором содержится название (value="**Belarus**") страны, в стране содержится город (<sity>) с параметром в котором содержится название (value="**soligorsk**") города, в городе содержится адрес (<adress>), в котором перечислены все данные адреса, и кто проживает по данному адресу. Но эти данные ничего не делают. Это чистая информация, завернутая в теги. Для того, чтобы воспользоваться этой информацией, программист должен написать программу. Для того, что бы программа знала к какому элементу в документе нужно обратиться, она должна знать путь к этому элементу. Этот путь называется **XPath**

# Что такое XPath

Xpath — это язык запросов к элементам XML или XHTML документа. После того как мы применили XPath, нам возвращаются выбранные им данные. Чтобы получить интересующие данные, необходимо всего лишь создать запрос, описывающий эти данные. Другими словами, XPath это путь к элементу или списку элементов на странице.

## Использование XPath в разработке программ

Например, программист пишет программу для отправки письма Ивану Ивановичу Иванову. Для того, что бы программа знала куда нужно отправить письмо, мы должны указать ей путь. Например: бы хотим, что бы письмо дошло только до квартиры Ивана Ивановича, но не передавалось ему.

Для того, чтобы например почтальон доставил письмо до квартиры, ему потребуются след данные

continent => hart\_of\_world => countrie => sity => adress (данные адреса)=> man

Тоже самое и для программы - ей нужен путь для того, чтобы добраться до нужного элемента в документе. На языке XPath это будет выглядеть следующим образом

continent/hart\_of\_world/countrie/sity/adress/(данные адреса)/man

## Использование XPath в автоматизированном тестировании

При автоматизации тестирования используются XPath запросы к элементам HTML страницы.

***Автоматизированное тестирование*** это вид тестирования web-приложения, которое выполняться при помощи инструментов автоматизированного тестирования, например программа [XML2Selenium](http://xml2selenium.com). Тестировщик, пишет какой-то тестовый сценарий, для программы. Потом запускает ее, и программа выполняет тестирование сайта сама, по тестовому сценарию.

Пример:

Мы тестируем сайт туристических путевок. Нам нужно проверить, что заказы с разными наборами услуг обрабатываются правильно. Если мы будем делать это вручную, есть вероятность того, что мы забудем протестировать какую-нибудь комбинацию с набором услуг.

**Например**: выбрать путевку со след. набором услуг:

-**отель:** 5 \*

-**питание:** все включено

-**количество ночей:** 10

Находим все туры, которые соответствуют нашим данным.

1) нужно проверить все эти услуги, но только для 4\*отеля потом для 3\* и т.д.

2) нужно найти все туры в которых есть только завтрак, и опять для 5\*, 4\* и т.д.

3) и т.д.

/>

Для того чтобы проверить как можно больше комбинаций, пишем программе тестовый сценарий , в котором прописаны все возможные наборы услуг. Теперь программа будет тестировать все ситуации, которые мы ей указали.

Таким образом программа будет запускаться по определенному сценарию, и в случае ошибки проинформирует тестировщика или программиста.

**Поиск элемента по указанному XPath**

Пример:

Нам нужно написать тест, который проверяет отображение логотипа на странице. Откроем страницу, для которой мы пишем примеры (см. [Необходимые инструменты](#h.fli2cx14v1wf)). Что бы обратиться к элементу “логотип” на странице, нужно указать путь к этому логотипу.

Введем в строку поиска след XPath //div[@id='logo']/h1/a/img



В fireBug, подсветится тег (<img>) со всеми атрибутами, в котором содержится логотип. Надпись “1 matching node”, в нижней части окна firebug, означает, что на странице присутствует 1 узел который соответствует введенному XPath .

Как видно из рисунка, каждый элемент находится внутри элемента расположенного выше, и представляется в виде древовидной структуры.

/>

# Дерево документа

HTML-документ представлен в виде дерева элементов.

**Дерево документа** (document tree) — это схема построения документа, которая показывает связи между различными элементами страницы: порядок следования и вложенность элементов.

Пример:

<html>

<head>

<title>Заголовок страницы</title>

</head>

<body>

<div class="mainWrap">

<h1>Основной заголовок</h1>

<p>абзац текста.</p>

<ul>

<li>пункт 1</li>

<li>пункт 2</li>

</ul>

</div>

<div class="sideBar">

<h2>Второй заголовок</h2>

<p>Текст</p>

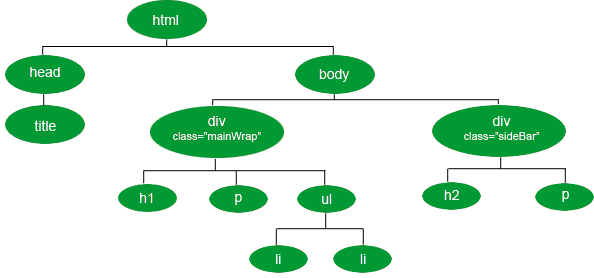
</div>

</body>

</html>

/>

Такой HTML код, пользователь может увидеть, если просмотрит код страницы. А вот если разобрать этот HTML по полочкам, то можно увидеть все уровни вложенности и взаимосвязи. Давайте разложим:

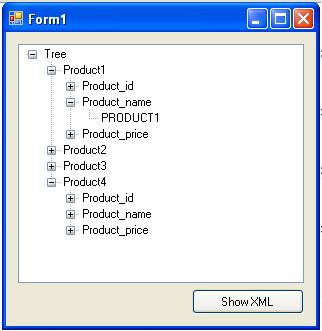


Мы выстроили из HTML кода, четкую иерархическую структуру в виде дерева (потому что схема похожа на очертания дерева). Как видно из рисунка, у дерева есть корневой элемент(root) тег (<html>), от которого отходят ветви, заканчивающиеся узлами.

## Узлы дерева

Узлами служат вложенные теги их атрибуты, тексты составляющие содержимое корневого элемента. От каждого вложенного тега, могут отходить свои ветви.

Узлы в дереве узлов имеют иерархические взаимоотношения друг с другом. Пример иерархических взаимоотношений. На рисунке отображается XML-документ в виде иерархии:



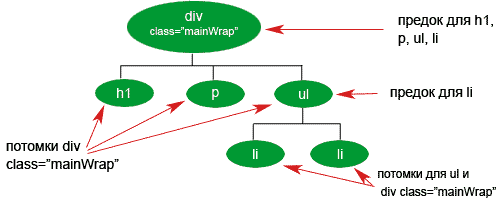
из рисунка видно, что в теге может находиться не только произвольное количество, но и вложенные в него теги. Таким образом вложенность тегов, образуют между собой родственные связи

## Родственные связи

### Предки и потомки

Из [схематического изображения дерева](#id.xwddmukiye88), да и из самого [HTML кода](#id.6saatiizaxpn), понятно, что одни элементы являются вложенными в другие. Элементы, которые содержат другие, являются **предками** (ancestor) по отношению к во всем вложенным в него. Вложенные в свою очередь являются его **потомками** (descendant).

Для наглядности рассмотрим одну ветку нашего дерева:



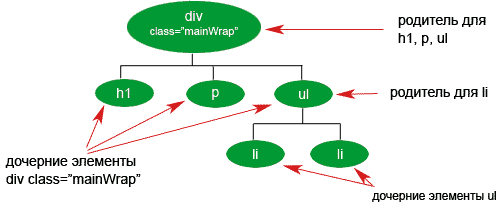
Каждый предок может иметь неограниченное число потомков. Каждый потомок будет иметь число предков в зависимости от структуры дерева и в какой ветке он будет расположен, но в любом случае как минимум один предок будет.

### 

### Родители и дочерние элементы

**Родитель** (parent) — это непосредственный предок (предок первого уровня) элемента. Пример из жизни: Отец(parent) является родителем (предком) сына.

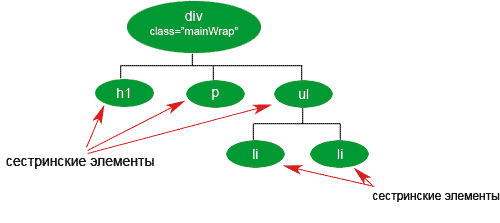
И наоборот, непосредственный потомок (потомок первого уровня) называется **дочерним элементом** (child). Пример из жизни: Сын (child) является дочерним элементом (потомком) Отца (patern).



Каждый родитель может иметь неограниченное число дочерних элементов. Пример из жизни: У отца может быть 5 и больше детей (Ваня, Петя, Коля, Оля, Галя….) Пример из жизни: У Вани может быть только один отец(биологический).

### Сестринские элементы

Сестринские элементы (siblings) — это группа из двух и более элементов, у которых общий родитель. Элементы не обязательно должны быть одного типа, просто у них должен быть общий родитель.



# Пути к элементам XPath

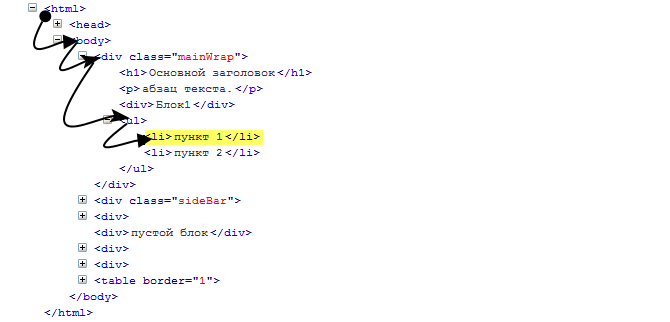
Все примеры, что приведены ниже, показаны на основе созданной нами [HTML-странички](http://xml2selenium.com/xpath/complex/).

Как уже говорилось [выше](#id.k9onvjdjz5ew) XPath это язык запросов к элементам html и XML страницы. Для того чтобы добраться к элементу используются пути. Они является наиболее полезным и широко используемым свойством XPath. Путь состоит из набора узлов XPath относительно его контекста.

Есть два вида путей *абсолютный* и *относительный*

## Абсолютный путь

*Абсолютный -* это путь от корня документа. Первым символом в нём должен стоять “/”. Корень документа всегда является узлом по умолчанию. Узел по умолчанию — это текущий полученный узел или набор узлов, относительно которых рассчитывается следующий шаг.



Что бы добраться до тега (<li>пункт 1</li>), нам нужно, начиная с корня документа (<html>) посетить каждый дочерний элемент родителя.

на примере с XPath это будет выглядеть следующим образом /html/body/div/ul/li[1]

Пример из жизни: есть 7 этажное здание. Что бы попасть на 7 этаж по лестнице, нам нужно посетить все этажи с 1-го по 7-ой - /1/2/3/4/5/6/7. Это будет абсолютный путь.

## 

## Относительный путь

*Относительный* - это путь от одного элемента (не обязательно от корневого) к другому. Чаще XPath-запрос начинают с «.//» или «//», это делает путь к элементу относительным. Символы "//" в начале запроса возвращают полное множество потомков, которые являются дочерними для корня документа, т.е. все элементы на текущей странице.

Например:



Чтобы добраться до тега (<li>пункт 1</li>), мы можем опустить все теги которые находятся выше тега (<div>) и заменить их на // и также можем заменить всех предков тега (<li>) на //, исключая тег (<div>).

На примере с XPath это будет выглядеть следующим образом: //div//li[1]

/>

Пример из жизни: есть 7 этажное здание. Нам нужно попасть с 3-го на 7-ой этаж, не посещая этажи 4,5,6. Для этого можно воспользоваться лифтом и пропустить этажи (4,5,6.). Наш путь будет выглядеть след. образом //3//7.

А если нам придется подниматься по лестнице, то наш путь будет выглядеть так

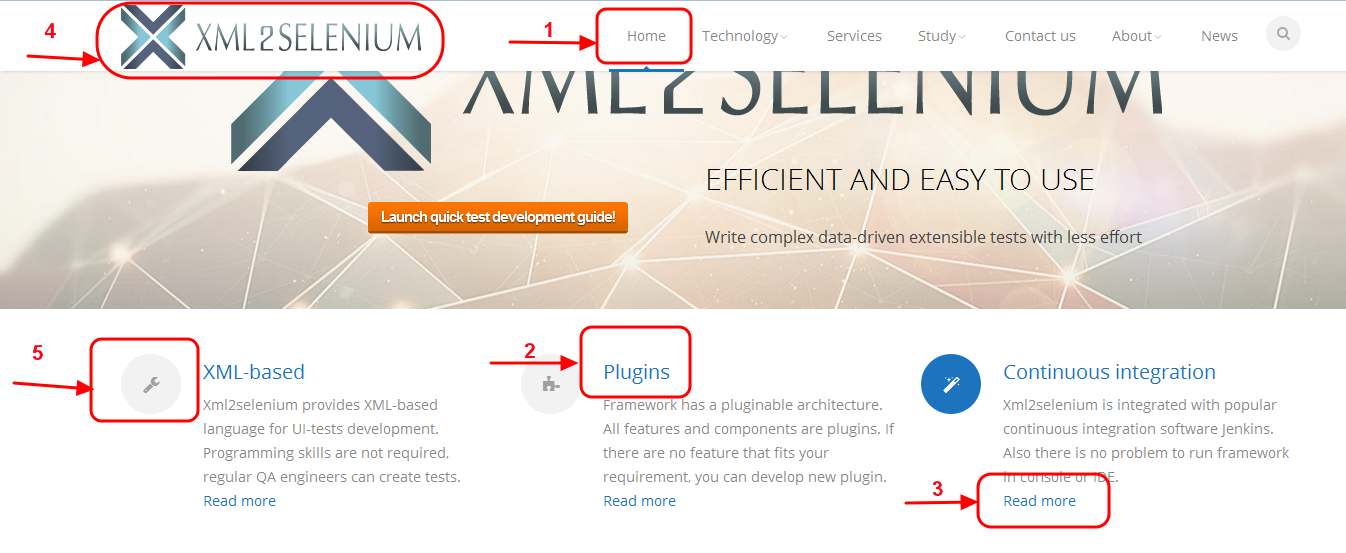
//3/4/5/6/7 Это будут относительные пути.

Задача:

Откроем страницу, для которой мы пишем примеры (см. [Необходимые инструменты](#h.fli2cx14v1wf)).

1. Верните все абсолютные пути, которые указаны на картинке.

2. Верните все относительные пути, которые указаны на картинке.

/>

# Синтаксис языка XPath

Для адресации частей документа выражение языка XPath использует обозначение пути, похожее на обозначение в URL-адресе.

Строка XPath — это фактически путь к элементу в дереве, где каждый уровень разделяется косой чертой «/». В результате обработки выражения XPath получается объект, который может быть:

***набор узлов***(node-set). Выражение /html/body/div вернет два узла элементов div, которые содержаться в элементе body.

***булево выражение*** (boolean). Выражение /html/body/div or p вернет значение true, т.к в элементе body содержится элемент div.

***числовой***(number). Если выбрать содержимое тега, содержащее число, то это число будет представлено числовым типом данных.

***строковый***(string). Выражение /html/body/div/h1['Основной заголовок'] вернет элемент h1 в котором содержится текст “Основной заголовок”, который содержится в первом элементе div.

Задача:

Откройте [HTML страничку](http://xml2selenium.com/xpath/complex/), и вставьте в строку для XPath пути, приведенные выше.

/>

Задание:

Откроем страницу, для которой мы пишем примеры (см. [Необходимые инструменты](#h.fli2cx14v1wf)).

1. привести 5 примеров для набора узлов.

2. привести 5 примеров для булевых выражений.

3. привести 5 примеров для числовых выражений.

4. привести 5 примеров для строковых выражений.

/>

XPath определяет два синтаксиса: *сокращенный* синтаксис и *не сокращенный* синтаксис.

## Сокращенный синтаксис:

Предоставляет набор символов. Ниже перечислены наиболее широко используемые символы:

### Дочерний элемент (/)

Выбирает дочерние элементы коллекции, указанной слева. Если этот оператор пути стоит в начале шаблона, будут выбраны дочерние элементы корневого узла.

Например:

/html/body/div/p

html/ выберет дочерний элемент body

body/ выберет дочерний элемент div и т.д.

/>

### Рекурсивный спуск (//)

Ищет указанный элемент на любой глубине. Используется для ссылки на все дочерними элементы узла контекста.

Т.е. если у нас есть такой путь //html/body/div/p, что бы работать с текстом, который расположен в теге (<p>), нам нужно знать только путь до родителя(<div>), а значит часть пути /html/body/ можно заменить на //.

Например:

//div ссылается на все элементы (<div>), которые есть на странице. Положение в иерархии не имеет значения, если (<div>) имеет вложенный элемент (<div>), то нам вернутся все элементы (<div>).

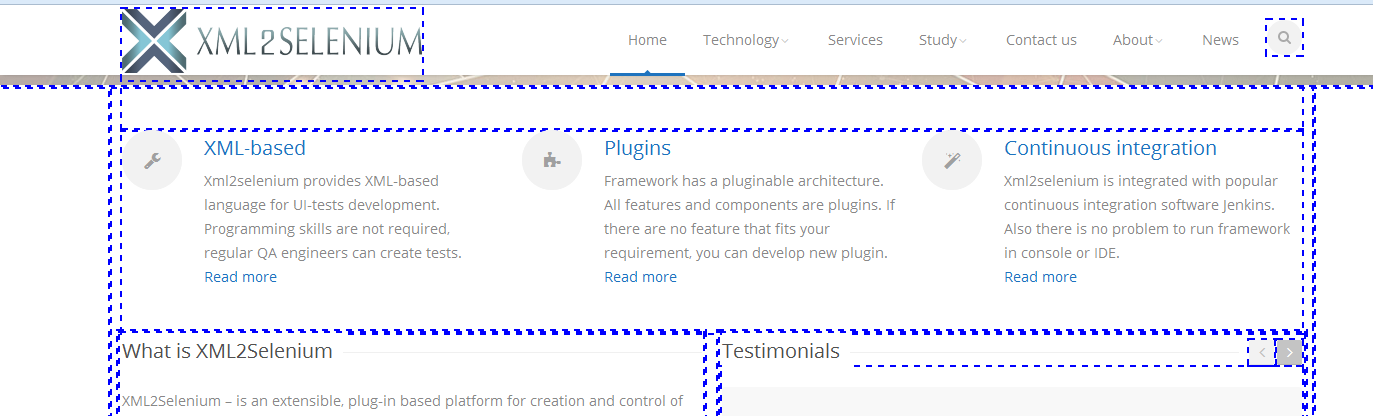
//div/p ссылается на все элементы (<p>), которые имеют родителя (<div>)

/>

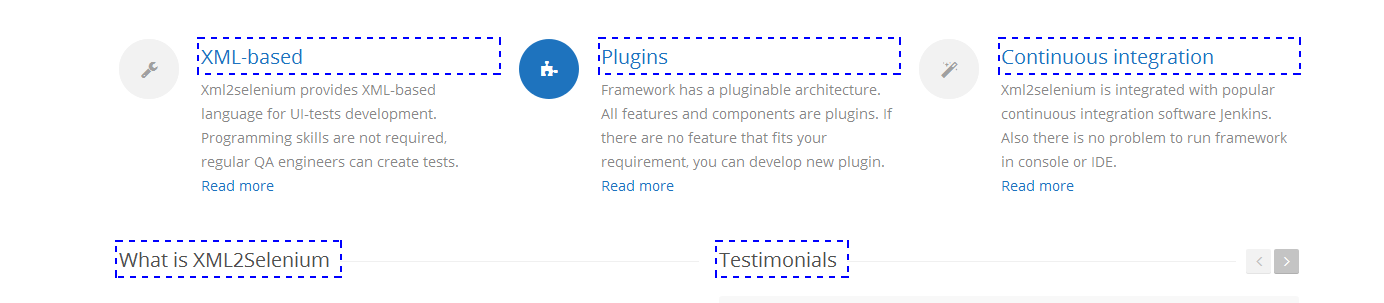
Задание:

Откроем страницу, для которой мы пишем примеры (см. [Необходимые инструменты](#h.fli2cx14v1wf)).

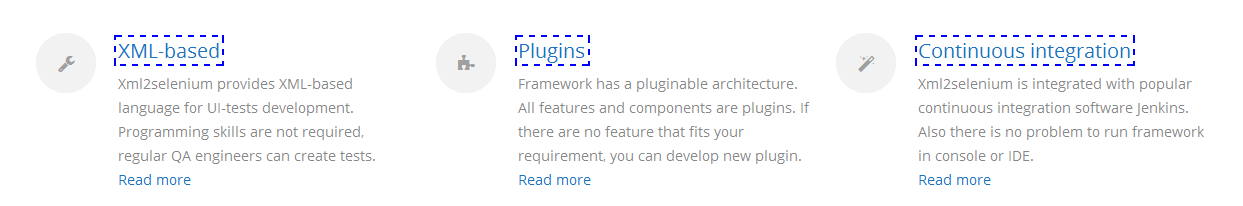
1.1 Выбрать все элементы (<div>) в котором содержится [атрибут](#id.mwyebqcjpk9l) @id;



1.2 Применить рекурсивный спуск до элемента (<h3>);

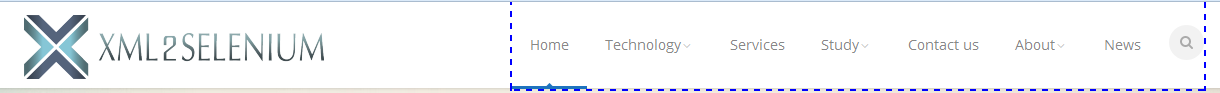


1.3 Выбрать дочерний элемент (<a>).

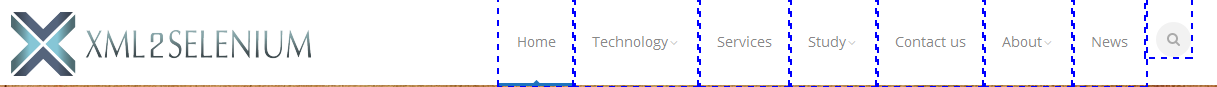


[ответ](#id.v3axujwmz2s7)

2.1 Обратимся к меню навигации, сделаем рекурсивный спуск до элемента (<nav>);



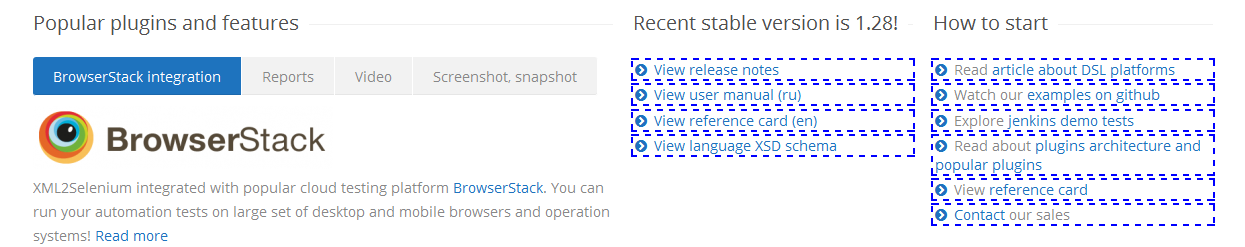
2.2. Обратимся ко всем элементам (<li>), сделаем рекурсивный спуск до элемента (<li>);



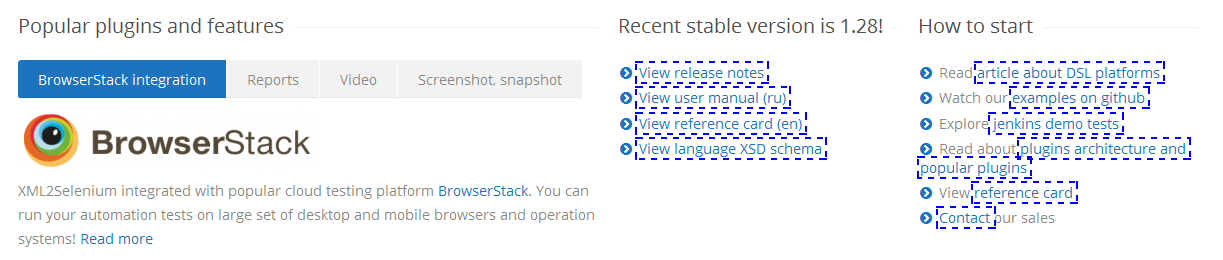
2.3 Выберем дочерний элемент (<a>) элемента (<li>).

[ответ](#id.v3axujwmz2s7)

3.1 Создать абсолютный путь до элемента (<li>);



3.2 Сделать рекурсивный спуск до элемента (<a>).



[ответ](#id.v3axujwmz2s7)

4. Придумать 2 XPath самостоятельно. Выбрать 2 понравившиеся картинки на сайте и составить XPath с рекурсивными спусками, например //div//tr/td//a

/>

### 

### Фильтрация ([])

Поскольку результатом выполнения запроса XPath бывает множество элементов, то результат называется *коллекцией*. Выражения XPath позволяют легко найти определенный узел/узлы документа в коллекции.

//div данный XPath вернет 8 элементов (<div>). Для того, чтобы можно было работать с любым элементом из коллекции нужно поместить в квадратные скобки номер элемента, с которым мы хотим работать.

//div[2] данный XPath вернет 2-й элемент (<div>) из коллекции элементов

Данные, заключенные в квадратные скобки называются *предикатами.*

***Предикаты*** — это фильтры. После названия узла в квадратных скобках можно указать условие фильтрации. Попадают те узлы, которые соответствуют условию фильтрации

Пример:

<table border="1">

<tr>

<td>

<p class="tr1">ячейка1</p>

</td>

<td>

<p class="tr1">ячейка2</p>

</td>

</tr>

<tr>

<td>

<p class="tr2">ячейка3</p>

</td>

<td>

<p class="tr2">ячейка4</p>

</td>

</tr>

</table>

//tr/td/p[@class='tr1']

По данному XPath находятся все элементы (<p>), затем вычисляется *предикат* (@class='tr1'), и элементы (<p>), которые не содержат значения (@class='tr1'), удаляются. После запуска нашего XPath на странице найдется 2 узла, которые соответствуют нашему XPath.

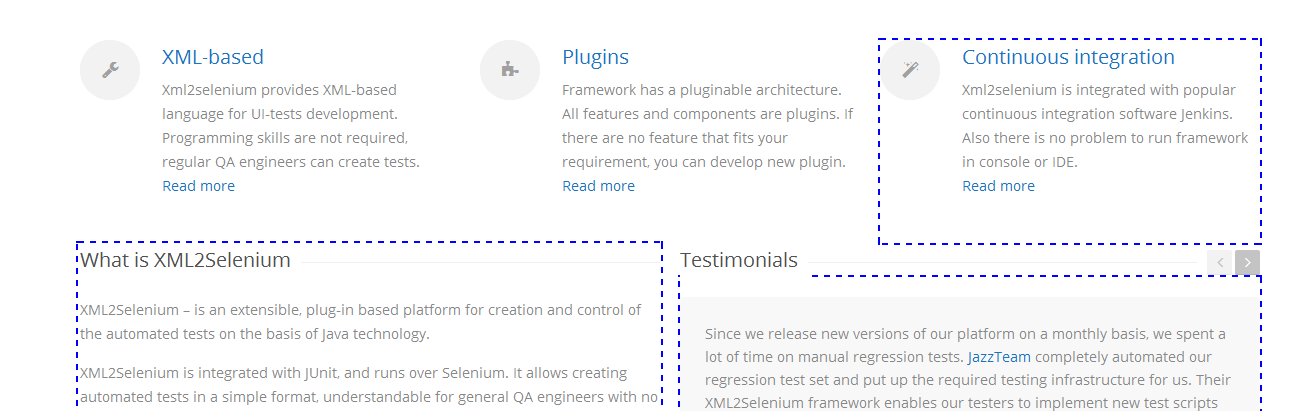
/>

Предикаты могут использовать также операции отношений >, <, >=, <= и !=. Они могут также использовать булевы операторы, которые будут рассмотрены позже

Задание:

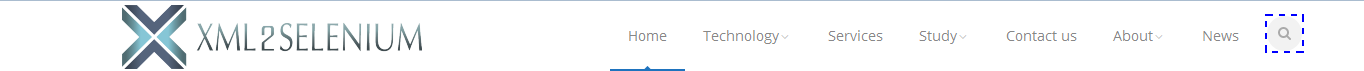
Откроем страницу, для которой мы пишем примеры (см. [Необходимые инструменты](#h.fli2cx14v1wf)). Напишем 5 XPath.

1. Выбрать все элементы (<div>) которые находятся на 3-й позиции.



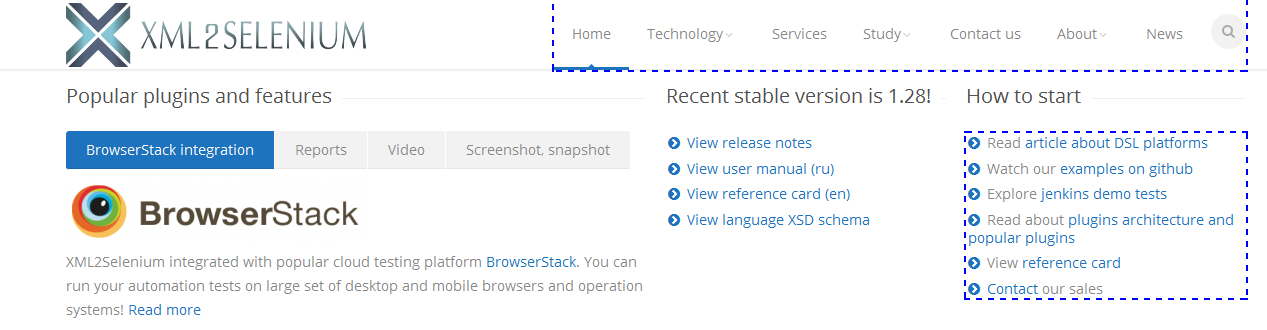
[ответ](#id.wb0lcdlrohgx)

2. Выбрать все элементы (<div>) на первой позиции, у которых есть дочерний элемент (<form>).

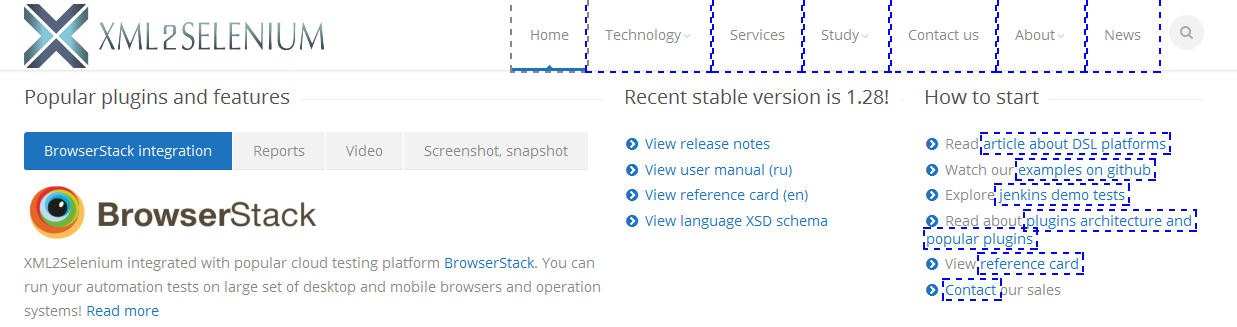


[ответ](#id.wb0lcdlrohgx)

3. Выбрать все элементы (<ul>), у которых дочерний элемент (<li>) находиться на пятой позиции.

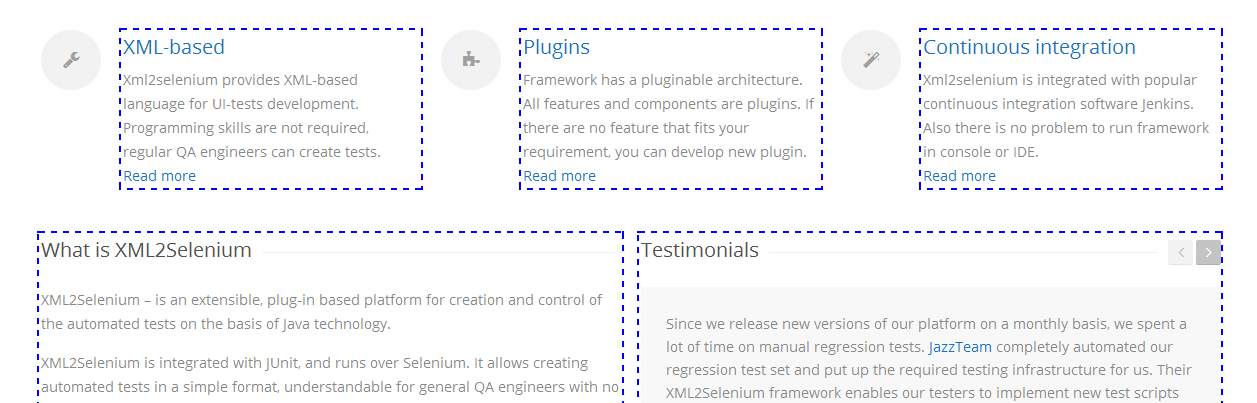


3.1 Сделать рекурсивный спуск к тегу (<a>).

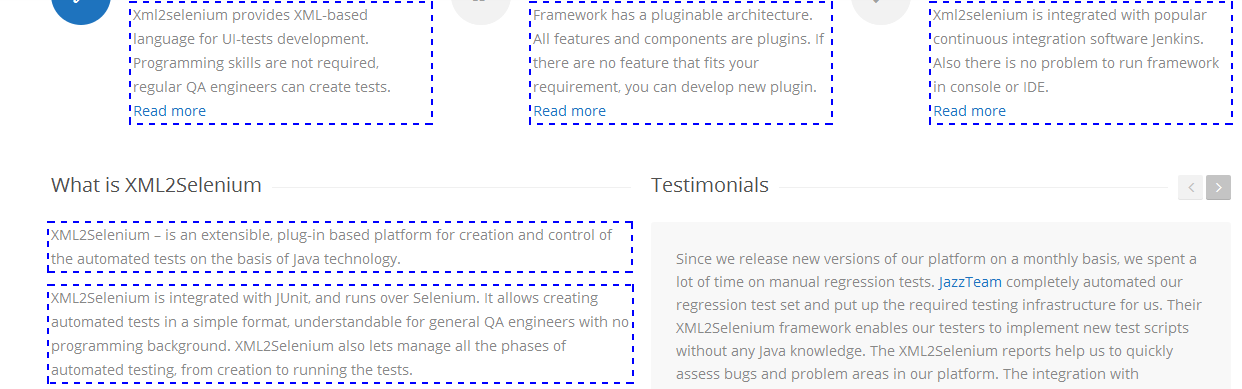


[ответ](#id.wb0lcdlrohgx)

4.1 Выбрать все элементы (<div>), у которых дочерним является тег (<h3>);

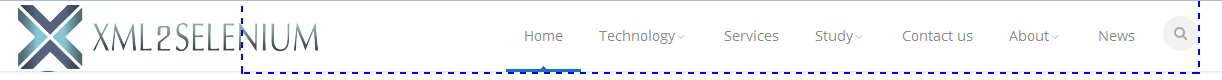


4.2 Выбрать дочерний элемент (<p>) тега (<div>).

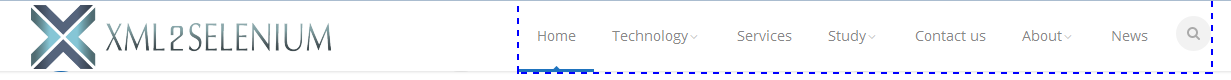


[ответ](#id.wb0lcdlrohgx)

5.1 Выбрать все элементы (<div>) которые находятся на первой позиции, в которых присутствует тег (<nav>);



5.2 Сделать рекурсивный спуск до тега (<ul>) в котором присутствует тег (<li>) который находится на 6-ой позиции;



5.2 Выбрать дочерний элемент (<а>).



[ответ](#id.wb0lcdlrohgx)

/>

Пример ещё одного XPath на эту тему: //div[//a]//tr[3]/td//

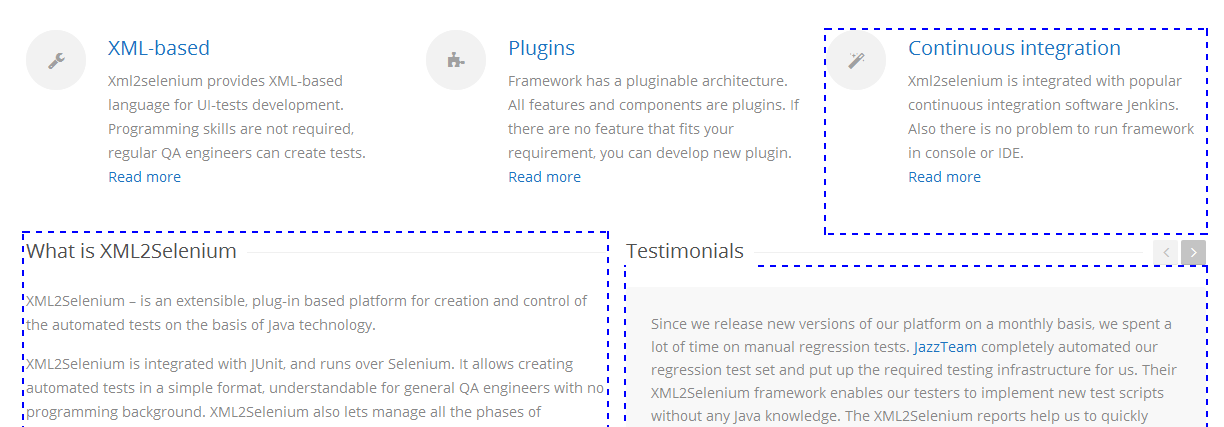
### Ссылка на все элементы (\*)

**\*** -используется для ссылки на все элементы, которые являются дочерними для узла контекста. Например: //\*/p ищет элемент (<p>) во всех элементах которые есть на странице. В примере выше //div/p мы искали p только в элементе (<div>).

Задание:

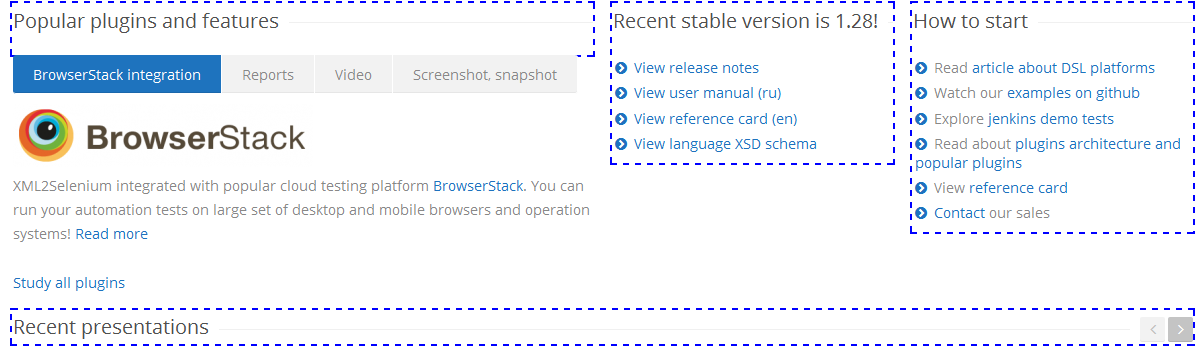
Откроем страницу, для которой мы пишем примеры (см. [Необходимые инструменты](#h.fli2cx14v1wf)).

1. Выбрать все элементы(\*), у которых дочерним является (<div>) и который находится на 3-й позиции.



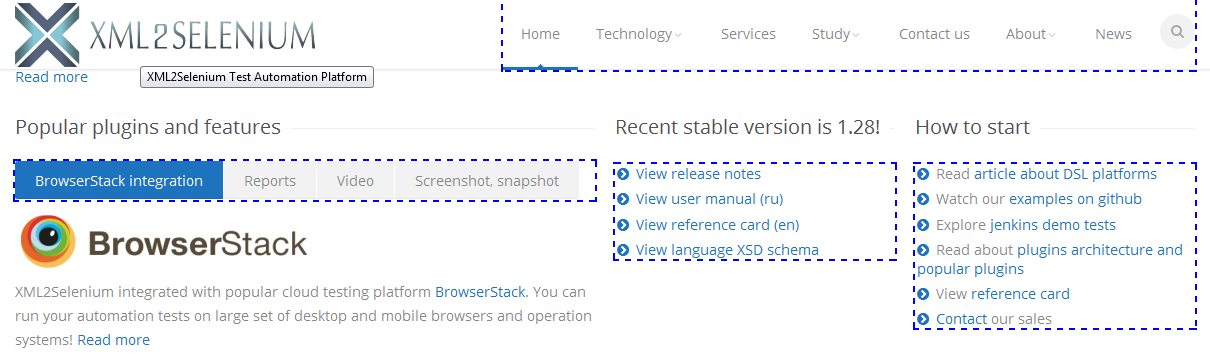
[ответ](#id.tnvyey5x36w)

2. Выбрать все элементы(\*), у которых присутствует тег (<h3>).

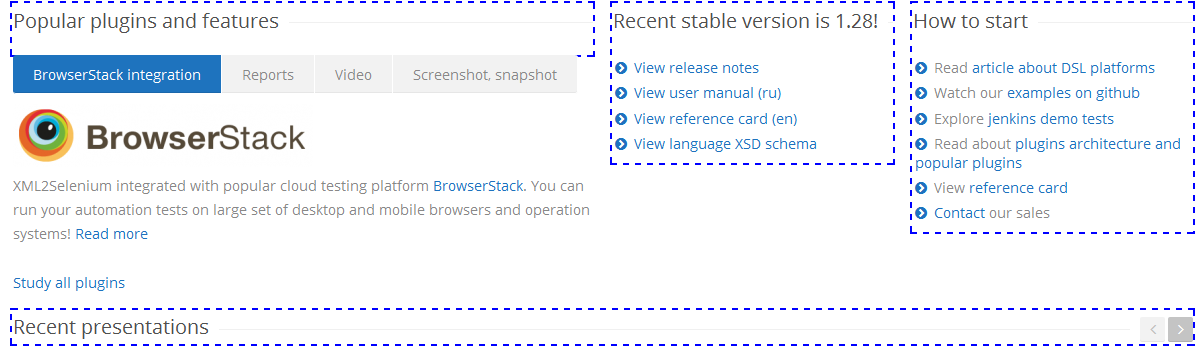


[ответ](#id.tnvyey5x36w)

3. Выбрать все элементы (<ul>).



3.1 Выбрать все элементы(\*), у которых дочерним является тег (<a>).



[ответ](#id.tnvyey5x36w)

/>

Пример использования XPath на эту тему: //\*[div]//\*/td/\*/a

### 

### Атрибут (@)

**@** - используется для ссылки на атрибуты.

Например: //div[@class='mainWrap'] ищет все элементы (<div>), в которых есть атрибут class со значением mainWrap

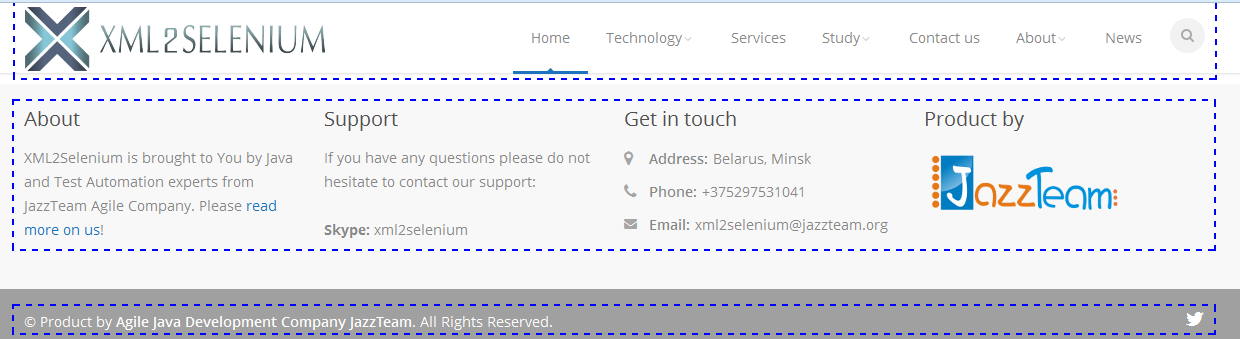
**@\*** -используется для ссылки на все атрибуты узла контекста.

Например: //div[@\*] ищет все элементы div в которых есть любые аттрибуты (id, name, style...)

Задание:

Откроем страницу, для которой мы пишем примеры (см. [Необходимые инструменты](#h.fli2cx14v1wf)).

1. Выбрать все элементы (<div>) в которых есть атрибут @class="container".



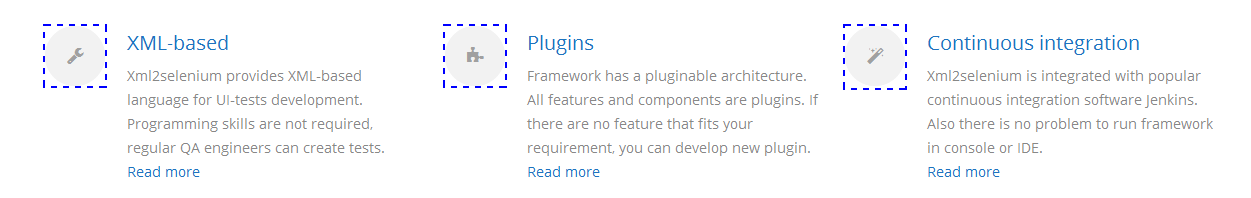
[ответ](#id.okr7ghri950o)

2. Выбрать все элементы (<div>) в которых атрибут @id.

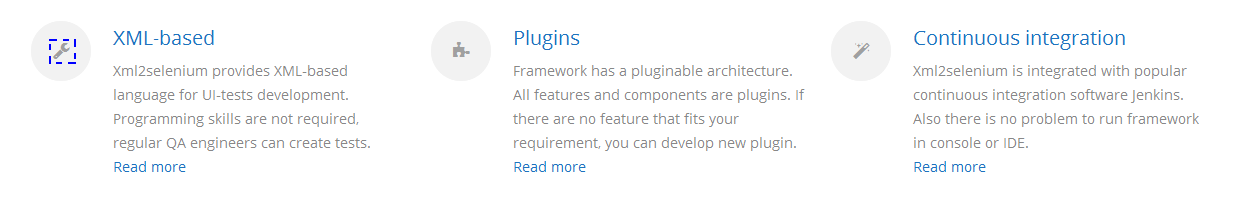


[ответ](#id.okr7ghri950o)

3.1 Выбрать все элементы (<div>) в которых есть атрибут @class='circle';

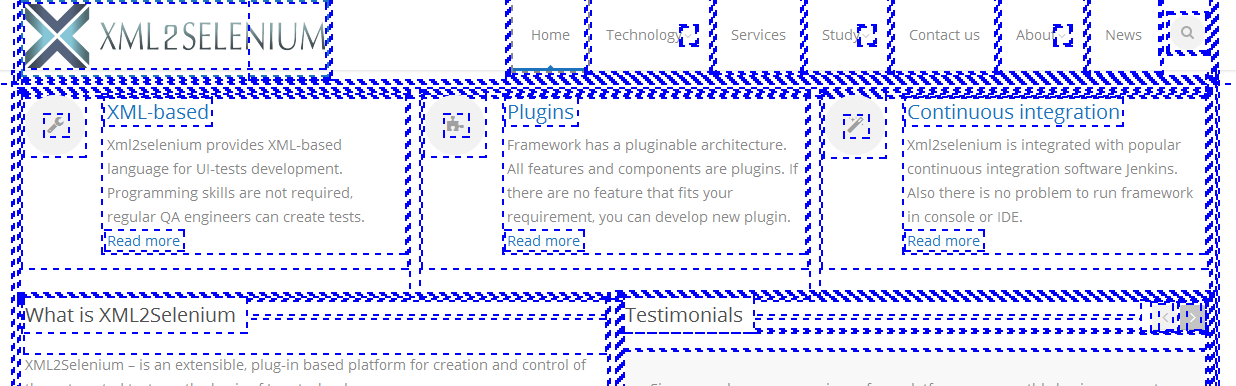


3.2 Выбрать дочерний элемент (<i>) с классом @class="icon-wrench".



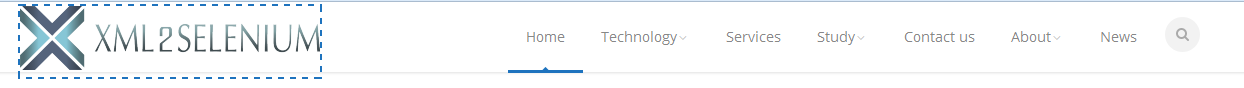
[ответ](#id.okr7ghri950o)

4.1 Выбрать все элементы (\*) в которых есть атрибуты (@\*);



4.2 Сделать рекурсивный спуск;

4.3 Выбрать все элементы (<a>), в которых присутствует атрибут @rel .



[ответ](#id.okr7ghri950o)

/>

Пример использования XPath на эту тему:

//div[@class="mainWrap"]/\*[@\*]//span[@name=”span”]

### 

### Ссылка на родительский узел (..)

.. используется для ссылки на родительский узел узла контекста.

Фрагмент кода:

<div>

<ul id="longNumber">

<li>35461</li>

<li>76893</li>

<li>95032</li>

</ul>

</div>

Например:

//div/ul[@id="longNumber"]/li[1]/../li[2]/../li[3]

//div/ul[@id="longNumber"]/li[1] данный XPath дойдет до первого li[1]

//div/ul[@id="longNumber"]/li[1]/.. вернется на один уровень выше. Этот XPath равен XPath //div/ul[@id="longNumber"]

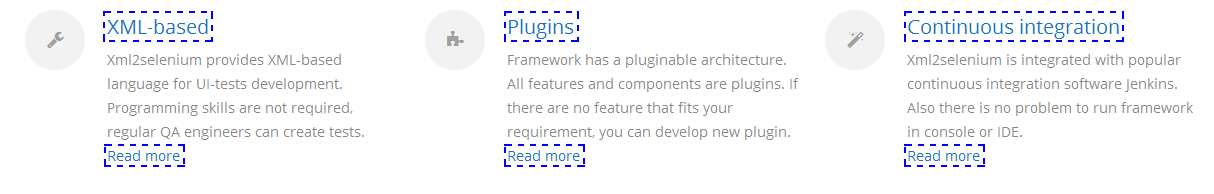
//div/ul[@id="longNumber"]/li[2] зайдет во второй li[2]

и т.д.

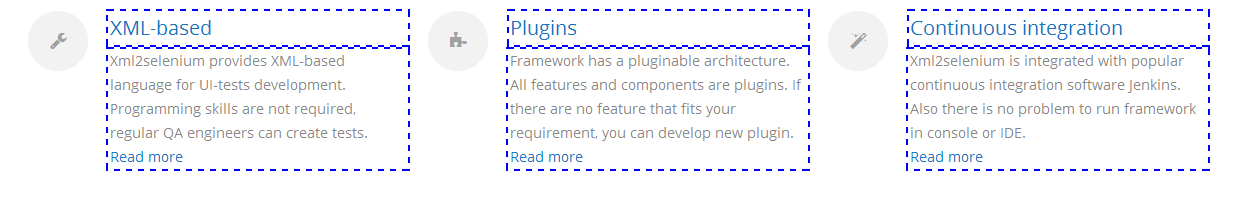
Задание:

Откроем страницу, для которой мы пишем примеры (см. [Необходимые инструменты](#h.fli2cx14v1wf)).

1.1 Выбрать все элементы (<a>);

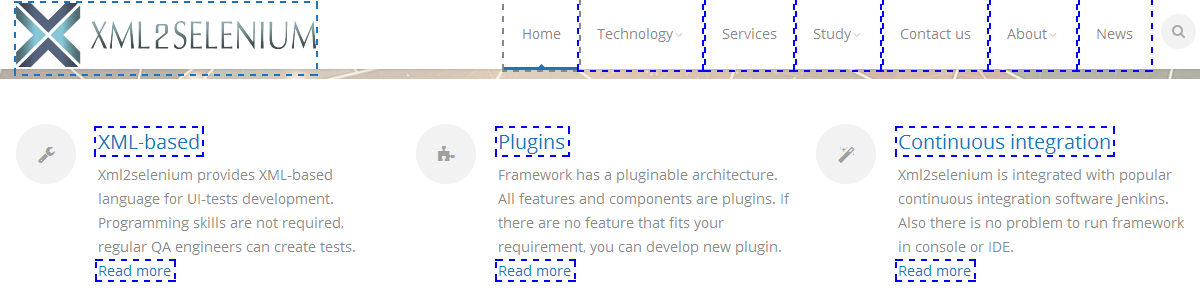


1.2 Вернуться на один уровень выше (..).

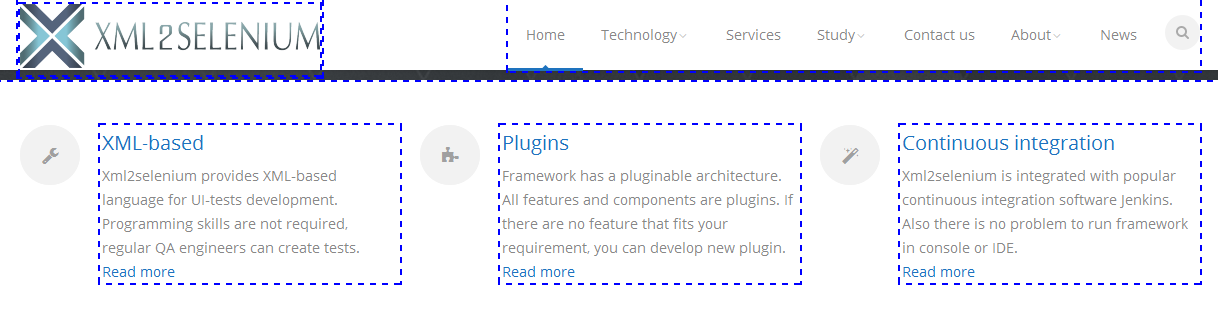


[ответ](#id.7bi3ufuibny3)

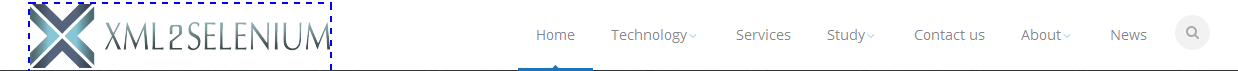
2.1 Выбрать все элементы (<a>);



2.2 Вернуться на два уровня выше;

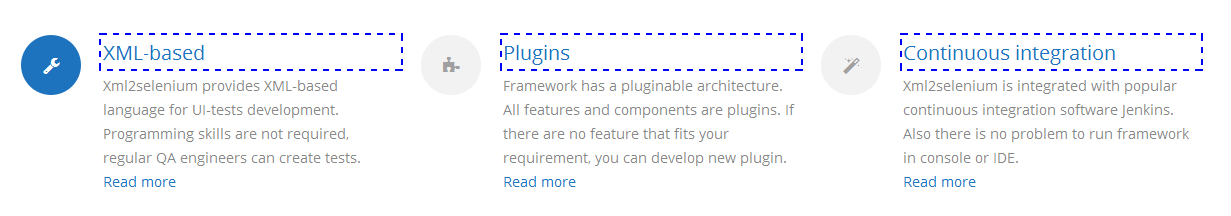


2.3 Выбрать элемент (<div>), в котором присутствует атрибут @id.

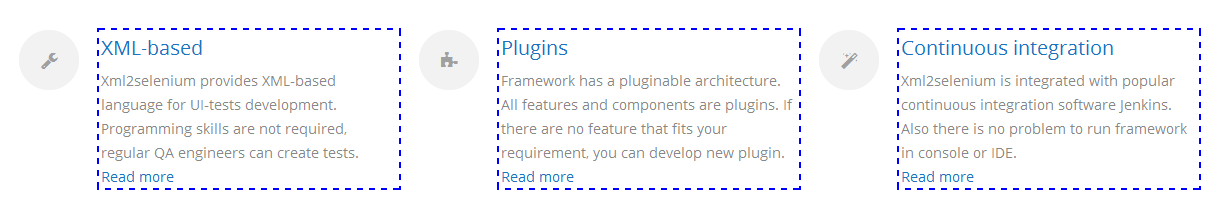


[ответ](#id.7bi3ufuibny3)

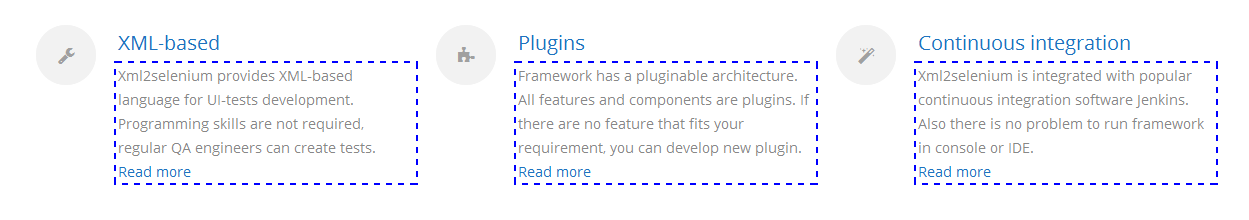
3.1 Выбрать все элементы (<h3>) у которых дочерним является тег (<a>);



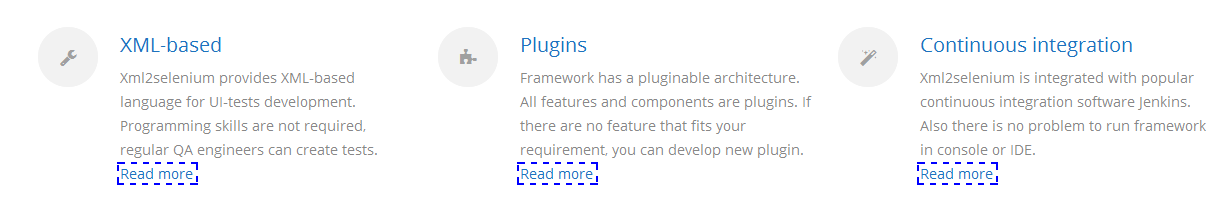
3.2 Вернуться на один уровень выше;



3.3 Выбрать тег (<p>);



3.4 Выбрать его дочерний элемент (<a>) .



[ответ](#id.7bi3ufuibny3)

/>

Пример использования XPath на эту тему: //div[@id=”id”]/../div[span]/ul/../ul

### 

### 

### Группировка операций ()

Группирует операции для явного задания порядка их выполнения.

Например:

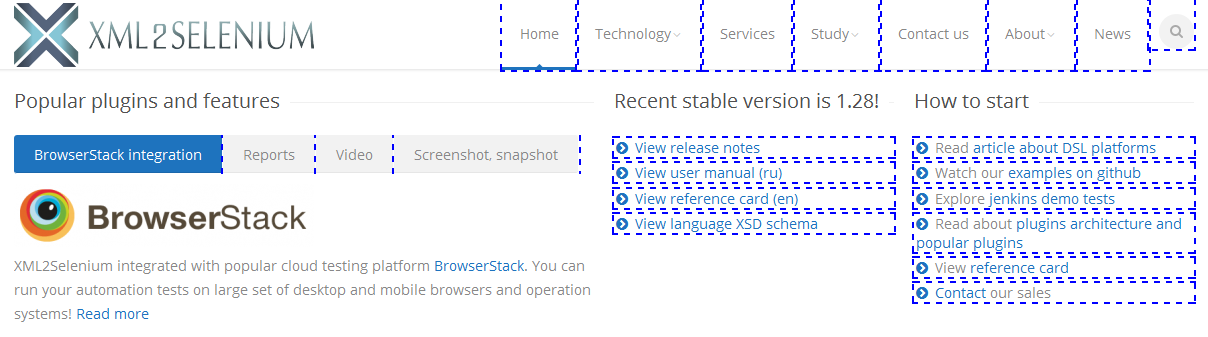
XPath //div[4] вернет все (<div>) которые находятся на 4-ой позиции

а XPath (//div)[4] ввернет 4-ый элемент из коллекции элементов, т.е. выражение (//div) создаст коллекцию из всех элементов (<div>).

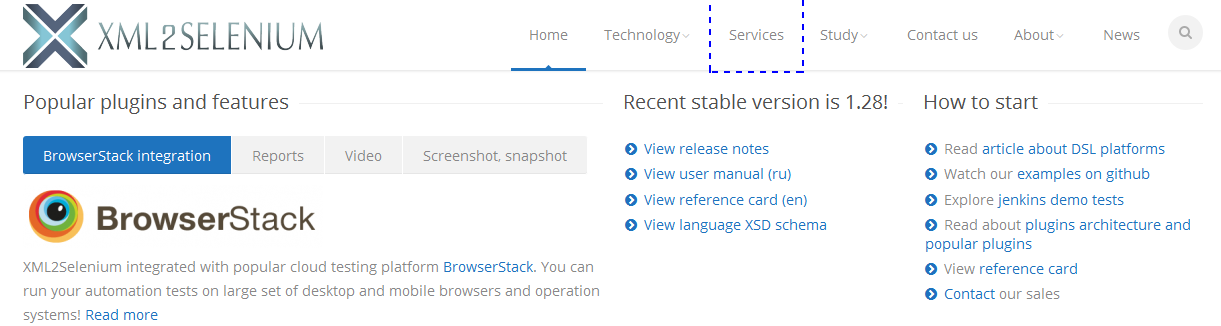
Задание:

Откроем страницу, для которой мы пишем примеры (см. [Необходимые инструменты](#h.fli2cx14v1wf)).

1.1 Сгруппировать все элементы (<li>);



1.2 Выбрать 10-ый элемент из коллекции.



[ответ](#id.9ptaq8kwvdr5)

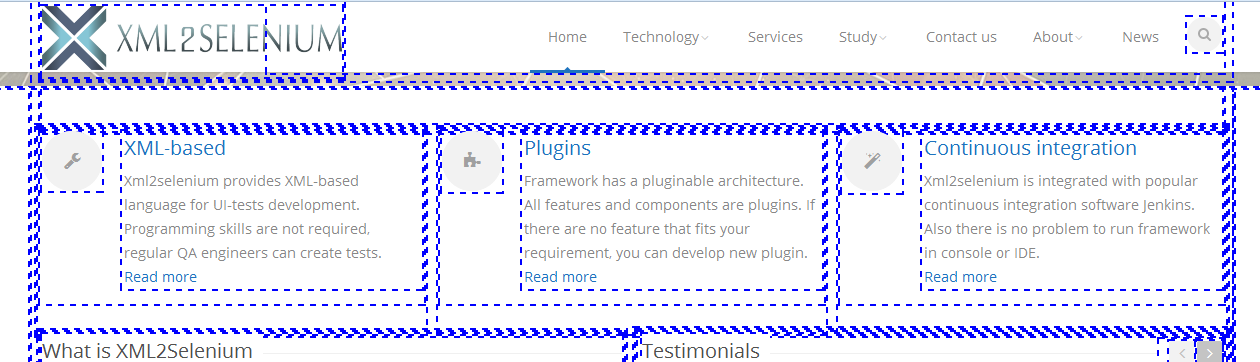
2. Придумать 3 XPath самостоятельно.

/>

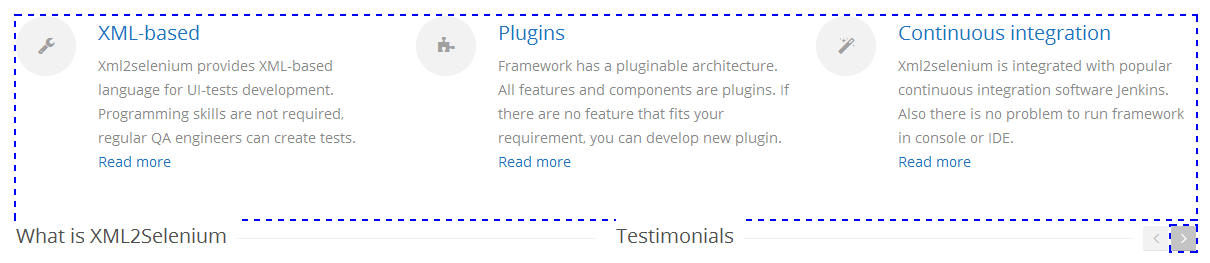
Общее задание:

Откроем страницу, для которой мы пишем примеры (см. [Необходимые инструменты](#h.fli2cx14v1wf)).

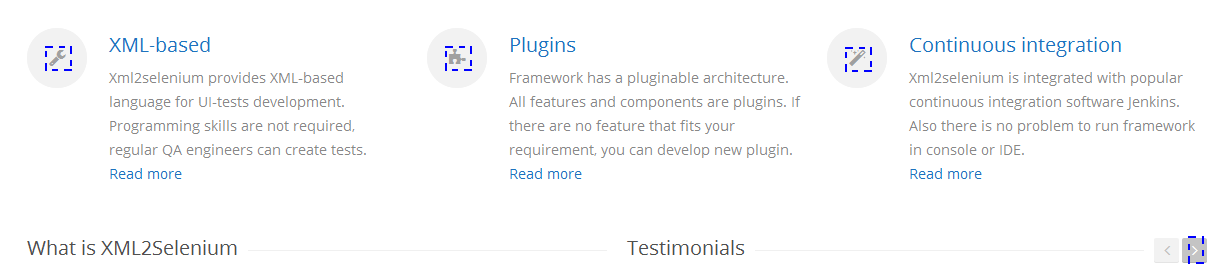
1.1 Выбрать все элементы (<div>);



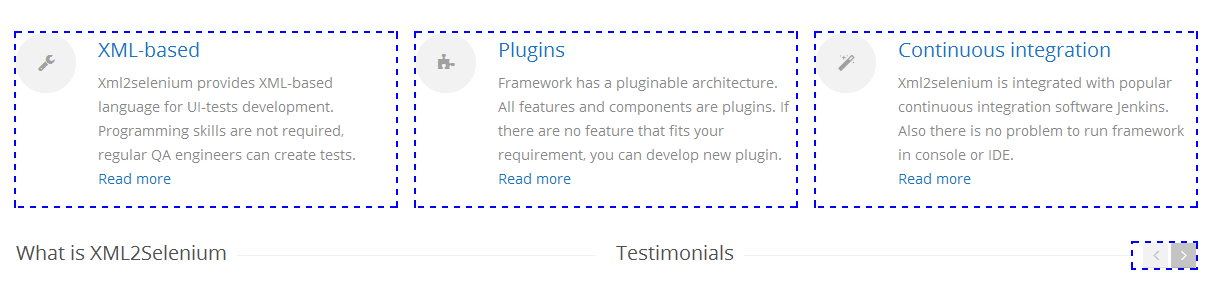
1.2 Выбрать дочерний элемент (<div>), который находится на второй позиции, у которого есть атрибут @id и атрибут @class;



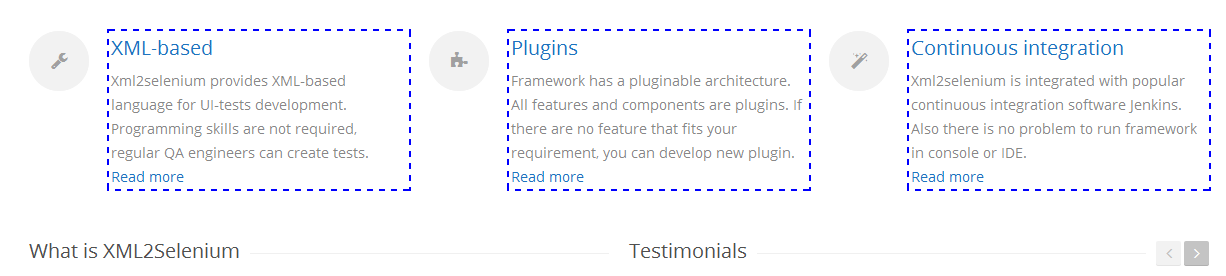
1.3 Сделать рекурсивный спуск к элементу (<i>);



1.4 Подняться на два уровня выше;

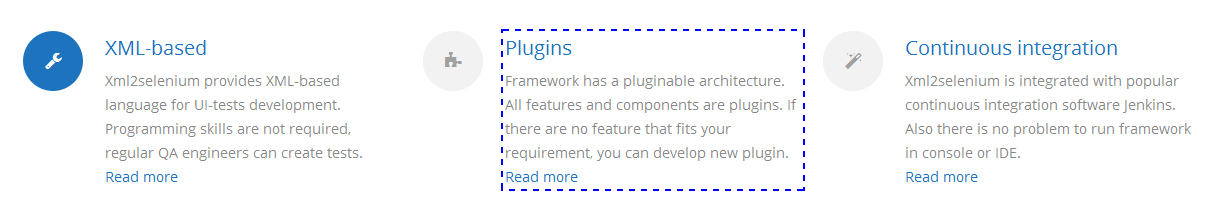


1.5 Выбрать тег (<div>), в котором есть тег (<h3>);

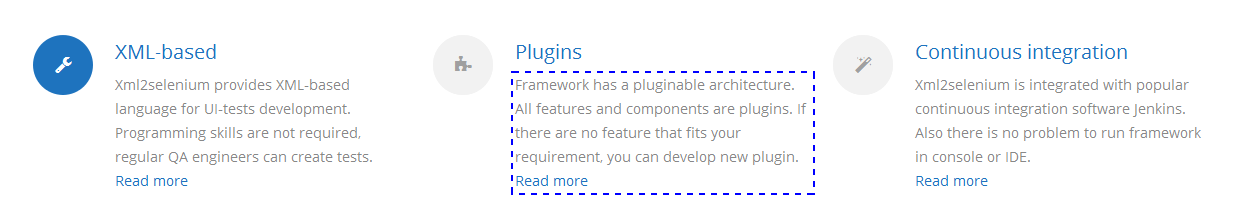


1.6 Сгруппировать все найденные элементы;

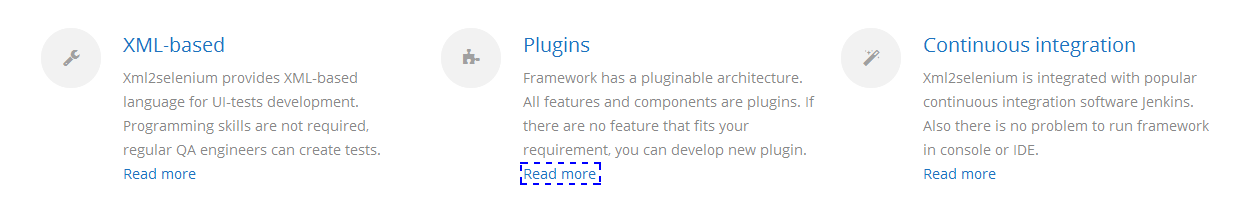
1.7 Выбрать 2-ой элемент из коллекции;



1.8 Сделать рекурсивный спуск к элементу (<p>);



1.9 Выбрать дочерний элемент (<a>) в котором содержится атрибут @href.



[ответ](#id.i4csazamwqu7)

/>

Пример использования XPath на эту тему:

//div[@class=”class”]/\*/../[@\*]/div[@name=”div”]

## 

## Не сокращенный синтаксис

### ancestor

**ancestor::** — возвращает множество предков.

Пример:

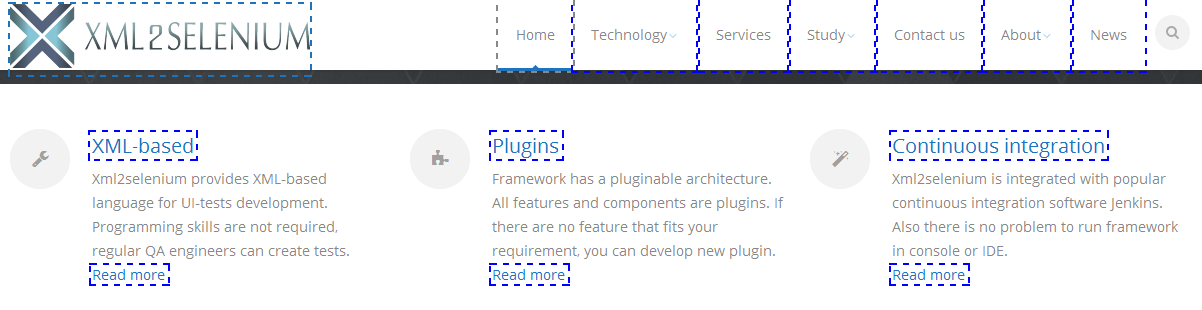


//ul[@id="number"]/li/ancestor::\* вернет всех предков тега (<li>)

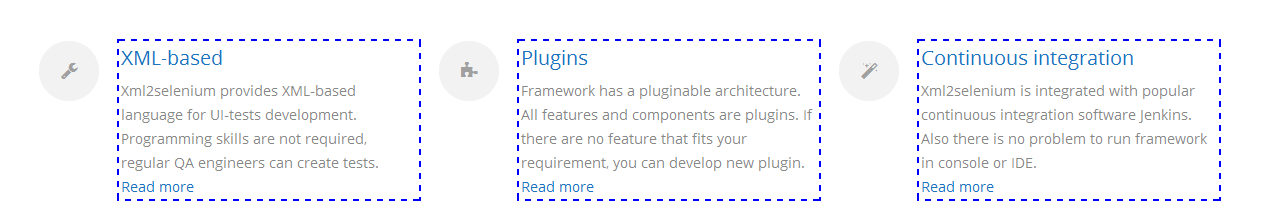
/>

Задание: Откроем страницу, для которой мы пишем примеры (см. [Необходимые инструменты](#h.fli2cx14v1wf)) и составим 5 XPath в которых присутствует ancestor::

1.1 Выбрать все элементы (<a>);

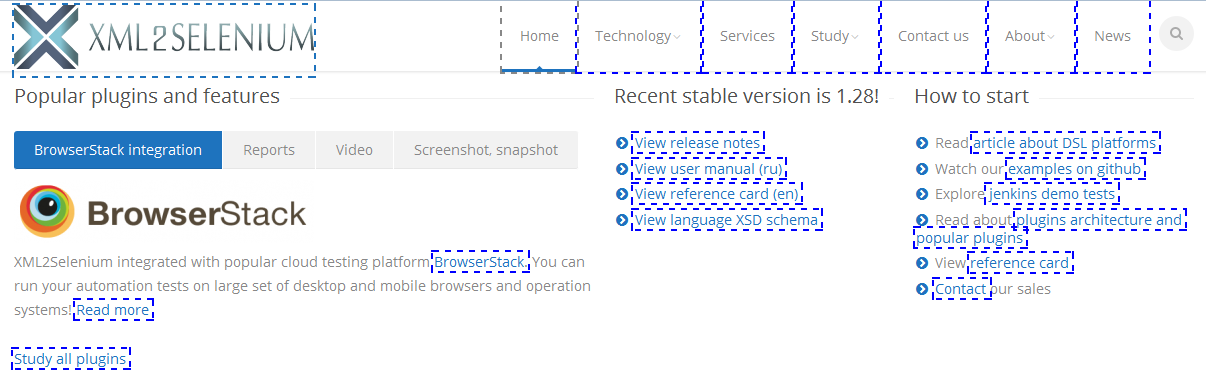


1.2 Вернуть всех предков (<div>) у которых дочерним является элемент (<h3>).

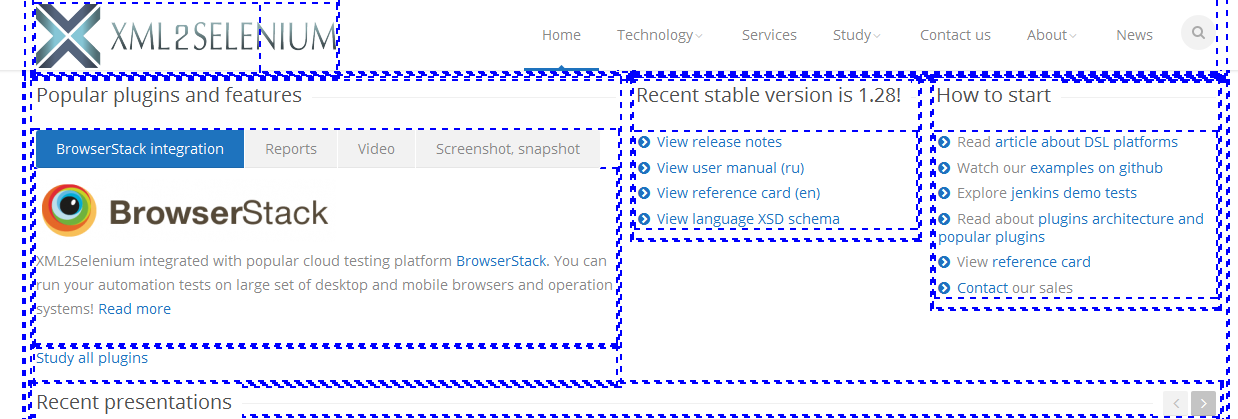


[ответ](#id.8h1tibuhvtrt)

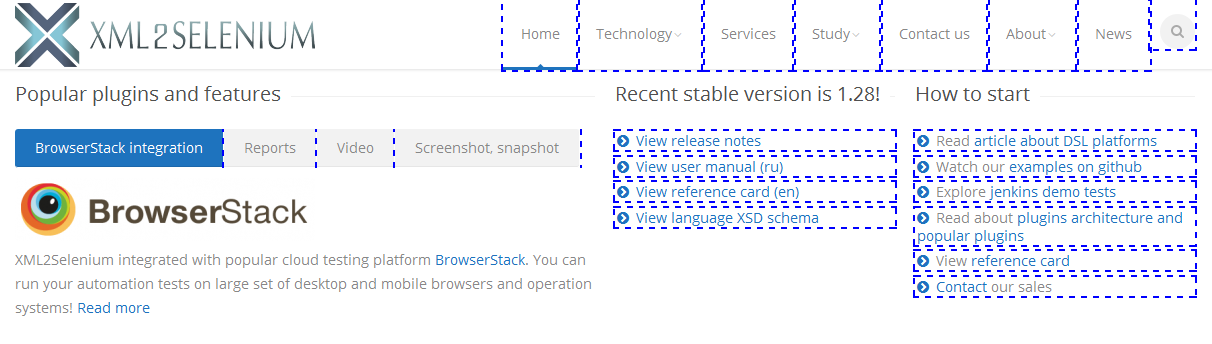
2.1 Выбрать все элементы (<a>);



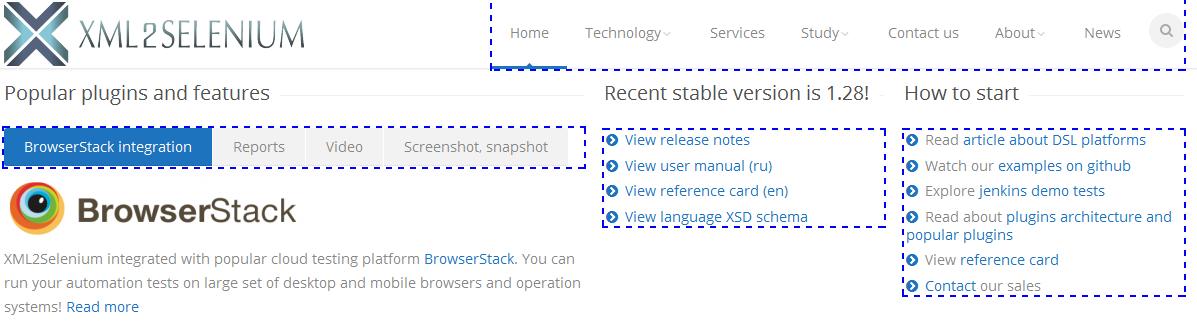
2.2 Вернуть всех предков (<div>);



2.3 Выбрать все элементы (<li>);



2.4 Вернуть всех предков (<ul>).



[ответ](#id.8h1tibuhvtrt)

/>

Пример использования XPath на эту тему:

//div[@class='b-line b-line\_bar']//a//ancestor::div//tr//ancestor::table

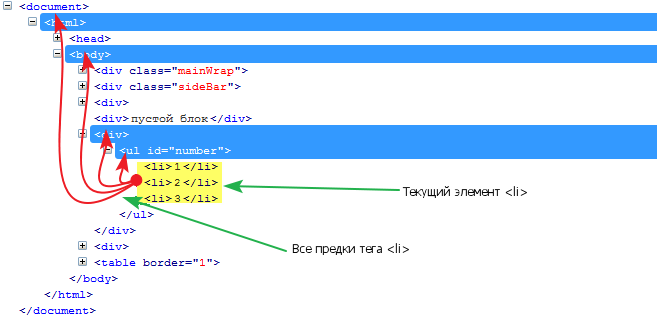
аттестат

### 

### ancestor-or-self

**ancestor-or-self::** — возвращает множество предков и текущий элемент.

Пример:



//ul[@id="number"]/li/ancestor-or-self::\* вернет всех предков тега (<li>) и сам тег (<li>)

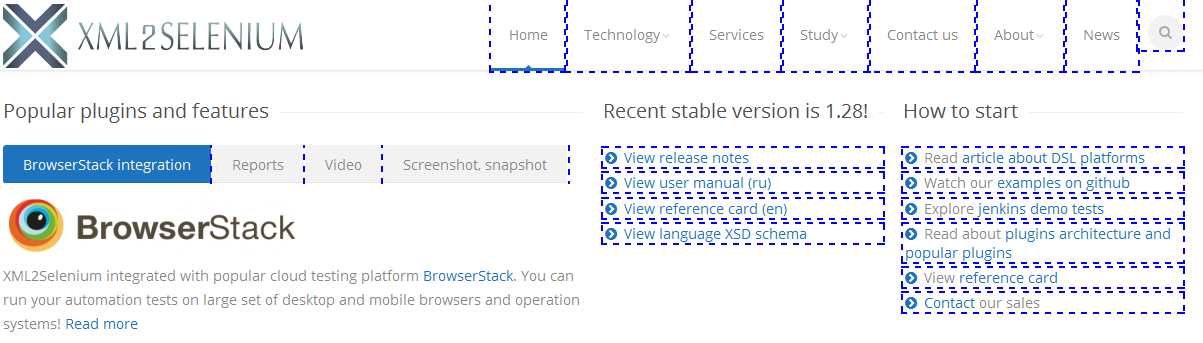
/>

Задание: Откроем страницу, для которой мы пишем примеры (см. [Необходимые инструменты](#h.fli2cx14v1wf)).

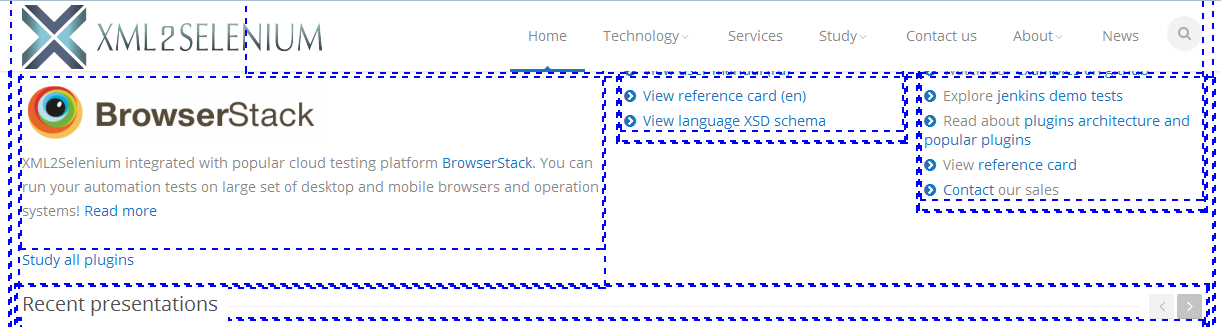
1.1 Выбрать все элементы (<ul>);



1.2 Выбрать дочерние элементы (<li>);



1.3 Вернуть всех предков (<div>), включая текущий элемент.

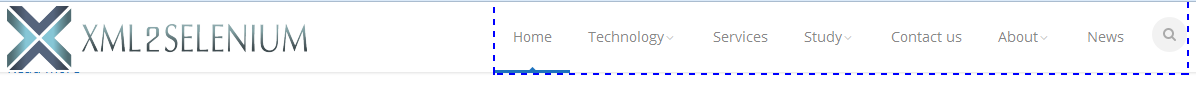


[ответ](#id.xbk9o0wp3fh8)

2.1 Выбрать все элементы (<a>);

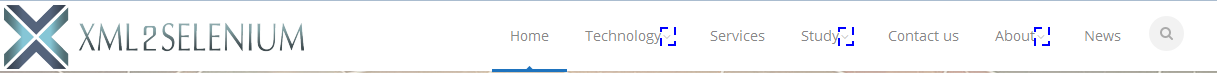


2.2 Вернуть всех предков (<ul>).



[ответ](#id.xbk9o0wp3fh8)

3.1 Выбрать все элементы (<span>), в которых есть элемент (<i>);



3.2 Выбрать всех предков включая текущий (<div>), котором содержится элемент.(<nav>)



[ответ](#id.xbk9o0wp3fh8)

### 

### attribute

**attribute::** можно заменить на **«@»** — возвращает множество атрибутов текущего элемента, //attribute::class вернет все элементы в которых содержится атрибут class

Задание: Откроем страницу, для которой мы пишем примеры (см. [Необходимые инструменты](#h.fli2cx14v1wf)) и составить 5 XPath в которых присутствует attribute::

1. Вернуть все элементы в которых есть атрибут class.

2. Вернуть все элементы в которых есть атрибут title.

3. Вернуть элементы с разными атрибутами.

[ответ](#id.kecrn69ne2gr)

/>

### child

**child::** — возвращает множество потомков на один уровень ниже.

Пример:

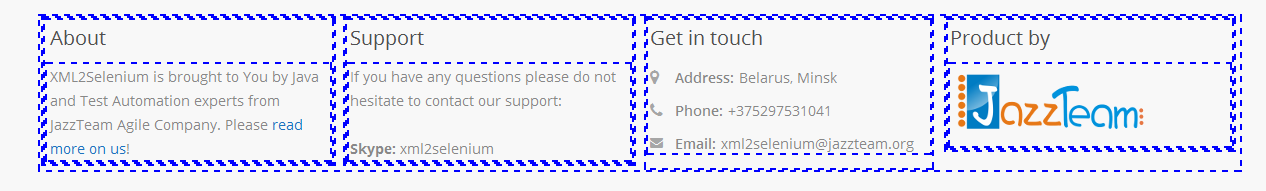
//body/child::div вернет все элементы (<div>)(потомки) которые содержаться в элементе (<body>)(предок)



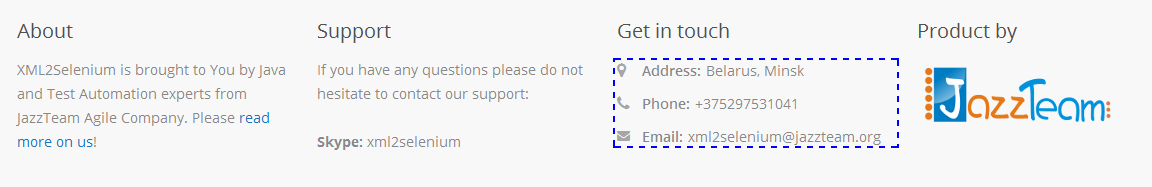
/>

Задание: Откроем страницу, для которой мы пишем примеры (см. [Необходимые инструменты](#h.fli2cx14v1wf)).

1.1 Выбрать все элементы (<div>);



1.2 Вернуть всех потомков (<ul>) элемента (<div>).



[ответ](#id.b31gy345ojd0)

2.1 Выбрать все элементы (<div>);



2.2 Сделать рекурсивный спуск;

2.3 Вернуть всех потомков (<div>) которые находятся на третей позиции.



[ответ](#id.b31gy345ojd0)

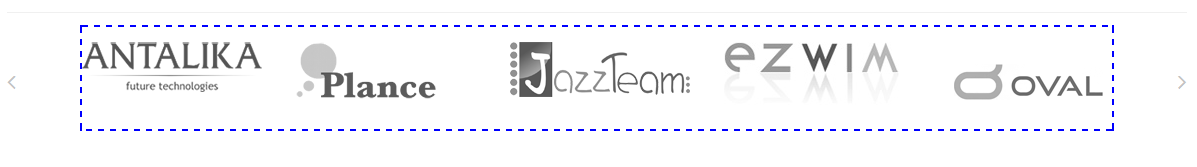
3.1 Выбрать все элементы (<div>);



3.2 Вернуть всех потомком (<div>), которые имеют атрибут @id;



3.3 Cделать рекурсивный спуск и выбрать все элементы (<div>) в которые присутствует элемент (<ul>).



[ответ](#id.b31gy345ojd0)

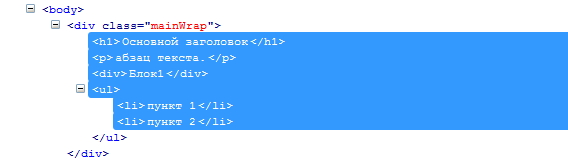
/>

### descendant

**descendant::** — возвращает полное множество потомков.

Пример:

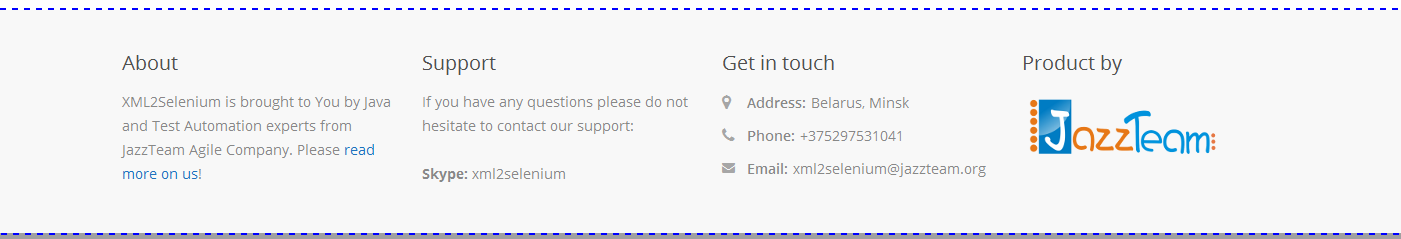
//div[@class="mainWrap"]/descendant::\* -вернет все элементы (потомки), которые содержаться в элементе (<div>)(предок).



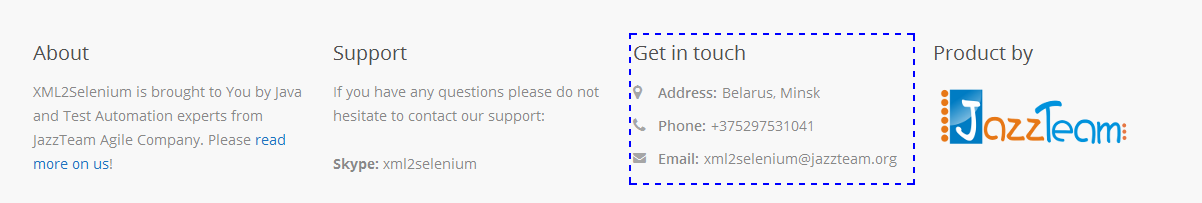
/>

Задание: Откроем страницу, для которой мы пишем примеры (см. [Необходимые инструменты](#h.fli2cx14v1wf)) и составить 5 XPath в которых присутствует descendant::

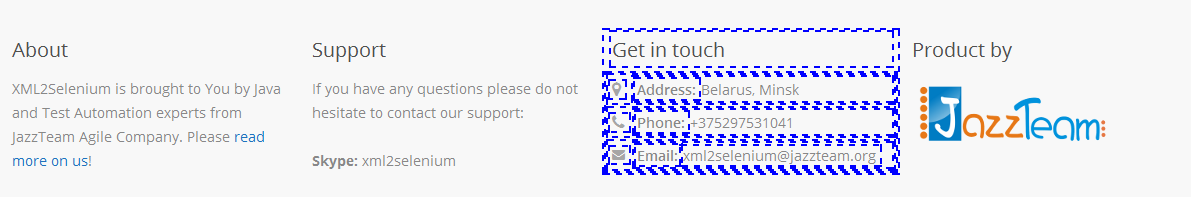
1.1 Выбрать все элементы (<div>) в которых есть атрибут id='footer';



1.2 Сделать рекурсивный спуск к элементу (<div>), который находится на 3-й позиции и у которого есть атрибут class='four columns' ;

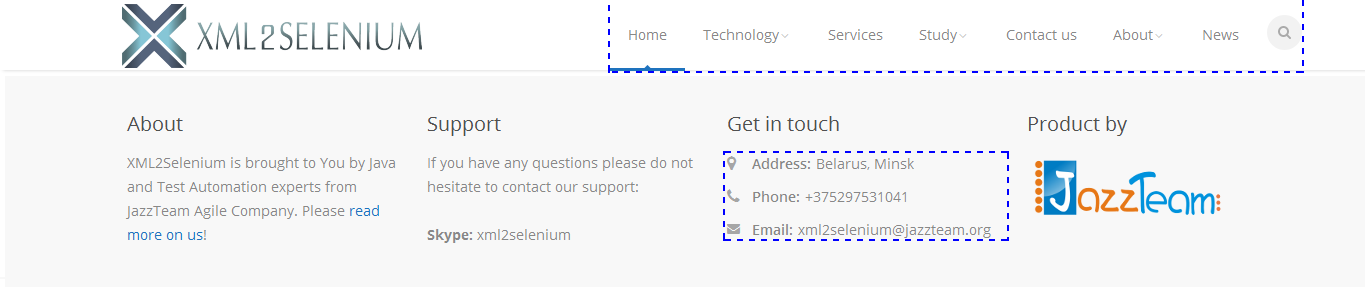


1.3 Вернуть всех потомков элемента (<div>).

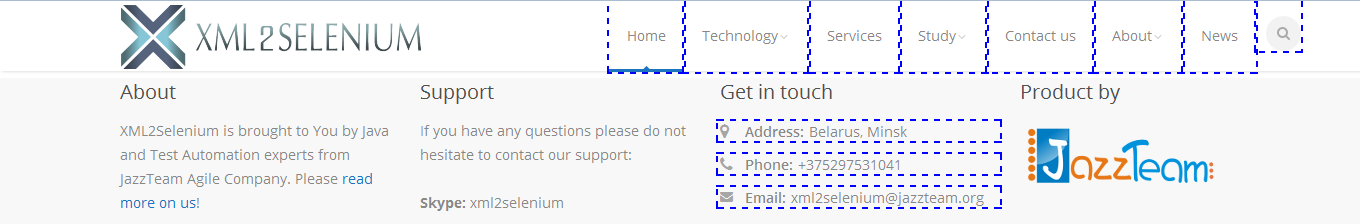


[ответ](#id.oqrvlweqsi1c)

2.1 Выбрать все элементы (<ul>);

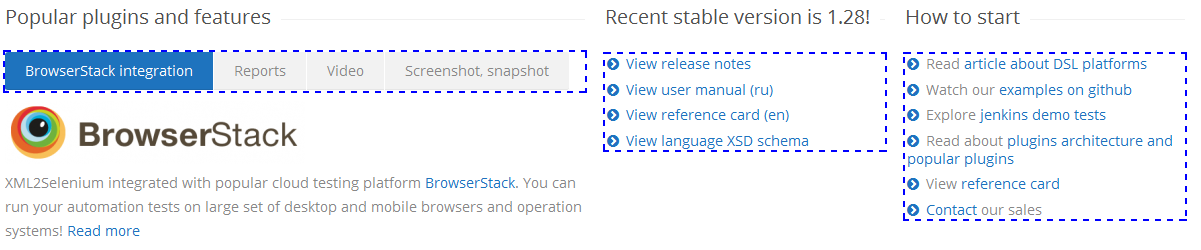


2.2 Вернуть всех потомков элемента (<ul>).

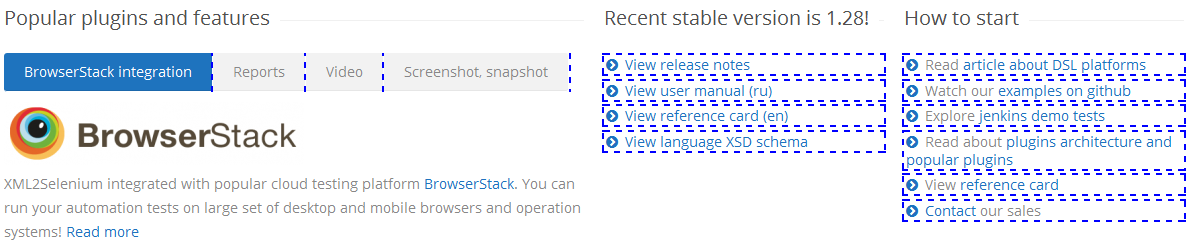


[ответ](#id.oqrvlweqsi1c)

3.1 Выбрать все элементы (<ul>);



3.2 Вернуть всех потомков (<li>) элемента (<ul>);



[ответ](#id.oqrvlweqsi1c)

4.1 Выбрать все элементы (<ul>), у которых предком является элемент (<nav>), и у которых есть элемент (<li>), который находится на 6-ой позиции;



4.2 Вернуть всех потомков (<a>) элемента (<ul>);



[ответ](#id.oqrvlweqsi1c)

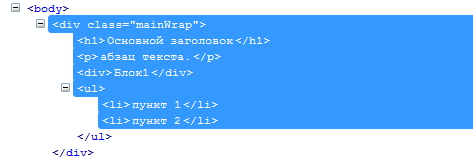
/>

### descendant-or-self

**descendant-or-self::** — возвращает полное множество потомков и текущий элемент.

Пример:

//div[@class="mainWrap"]/descendant-or-self::\* вернет все элементы которые содержатся в элементе (<div>), и сам элемент (<div>)



/>

Задание: Откроем страницу, для которой мы пишем примеры (см. [Необходимые инструменты](#h.fli2cx14v1wf)).

1.1 Вернуть всех потомков элемента (<div>) с атрибутом class=’\*\*\*’, и сам элемент (<div>).

2.1 Выбрать все элементы (<div>);

2.2 Вернуть всех потомков, включая текущий в которых содержится атрибут @style

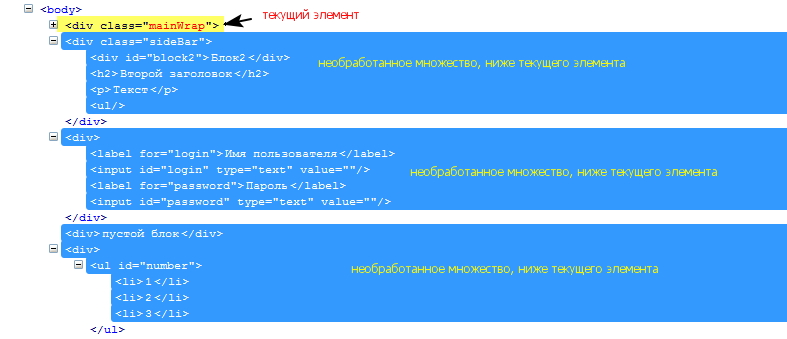
/>

### 

### following

**following::** — возвращает необработанное множество, ниже текущего элемента.

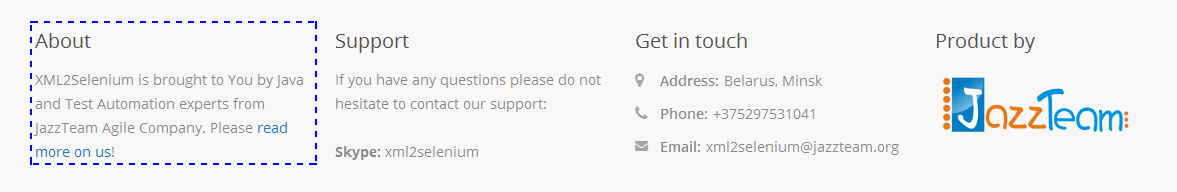
Пример: //div[@class='mainWrap']/following::\* вернет все элементы которые идут после элемента (<div[@class='mainWrap']>)



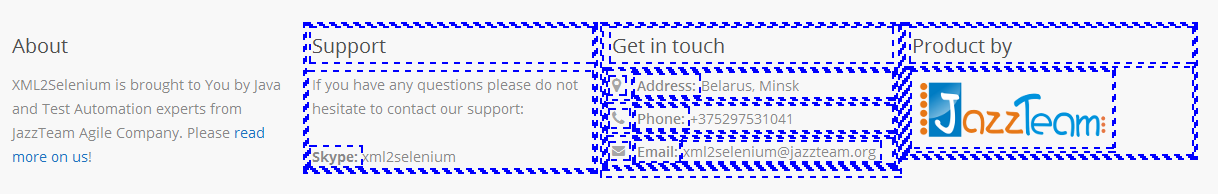
/>

Задание: Откроем страницу, для которой мы пишем примеры (см. [Необходимые инструменты](#h.fli2cx14v1wf)).

1.1 Dыбрать элемент (<div>) с атрибутом @id='text-2';

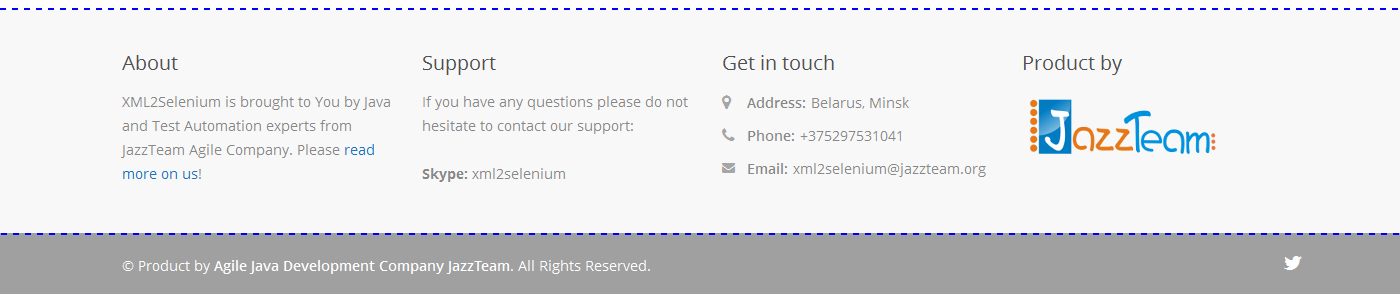


1.2 Вернуть не обработанное множество всех элементов ниже текущего;

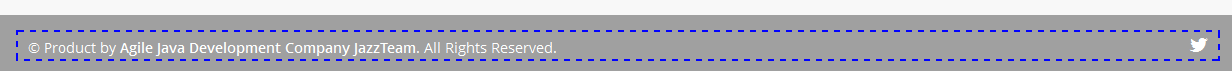


[ответ](#id.fgh3q2yujuah)

2.1 Выбрать элемент (<div>) с атрибутом @id='footer';



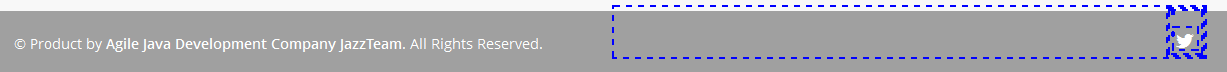
2.2 Вернуть не обработанное множество элементов (<div>) с атрибутом @class='container' ниже текущего;



2.3 Выбрать дочерний элемент (<div>) на первой позиции;

Image 1.png

2.4 Вернуть не обработанное множество элементов.



[ответ](#id.fgh3q2yujuah)

/>

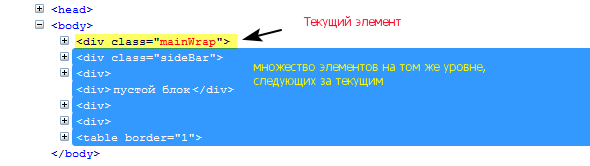
### 

### following-sibling

**following-sibling::** — возвращает множество элементов на том же уровне, следующих за текущим.

Пример: //div[@class='mainWrap']/following-sibling::\*

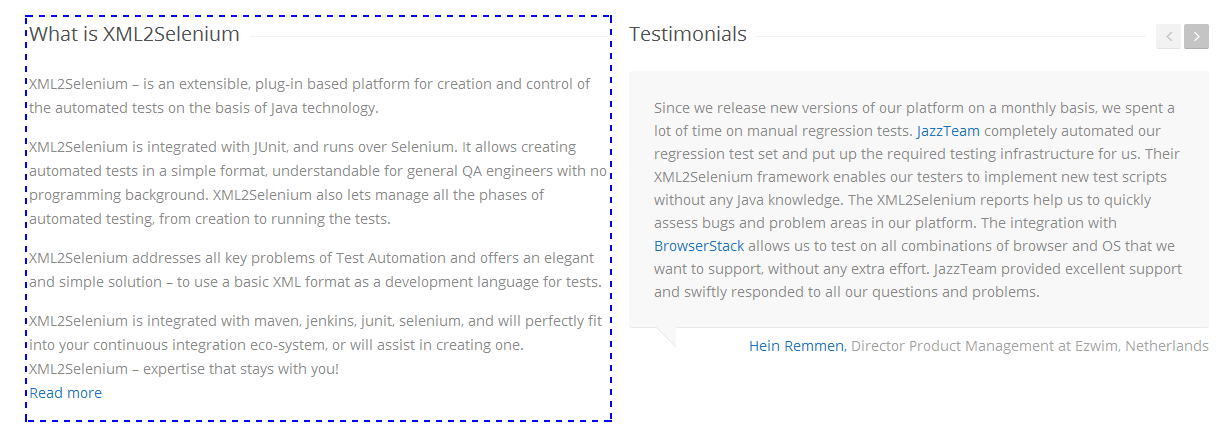
вернет все элементы(братьев и сестер) которые находятся на одном уровне с элементом (<div[@class='mainWrap']>)



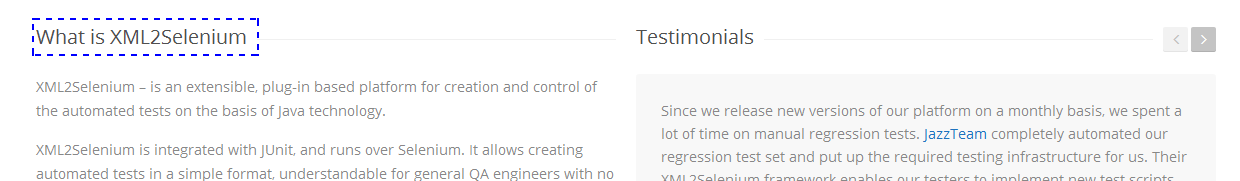
/>

Задание: Откроем страницу, для которой мы пишем примеры (см. [Необходимые инструменты](#h.fli2cx14v1wf)).

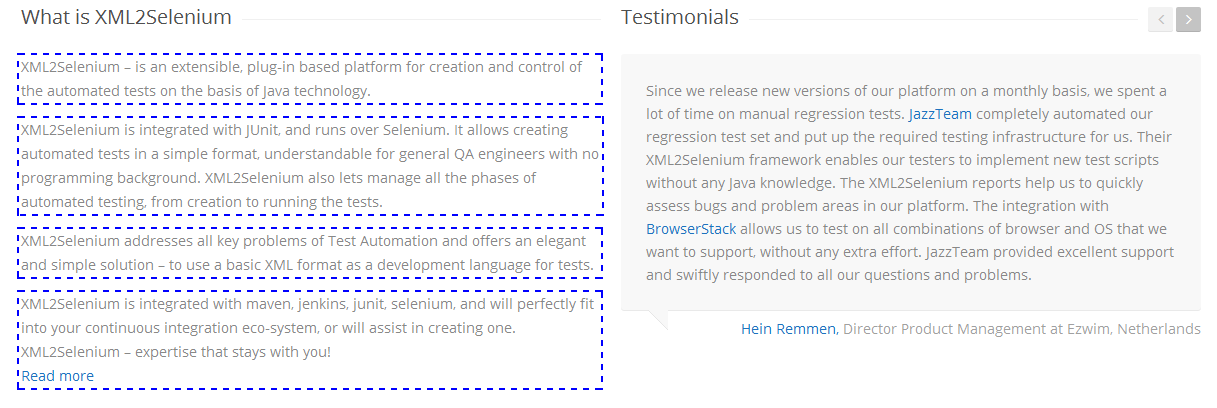
1.1 Выбрать все элементы (<div>), в которых есть тег (<p>) и тег (<h3>)с атрибутом @class="headline";



1.2 Выбрать тег (<h3>);



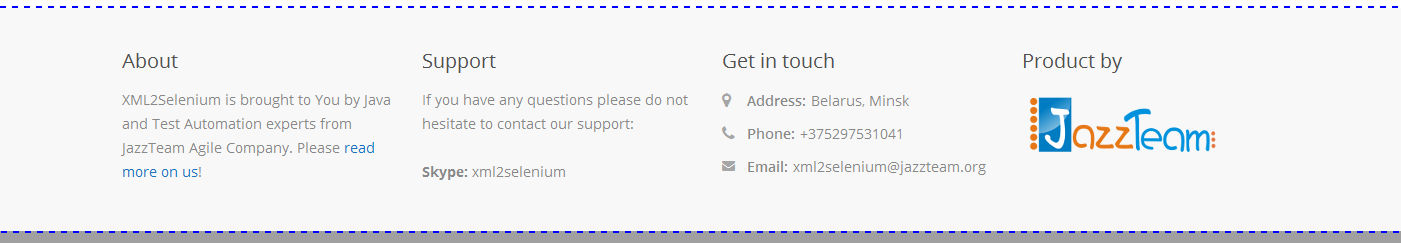
1.3 Вернуть множество элементов (<p>), на том же уровне что и тег (<h3>).



[ответ](#id.kquj905ohyn)

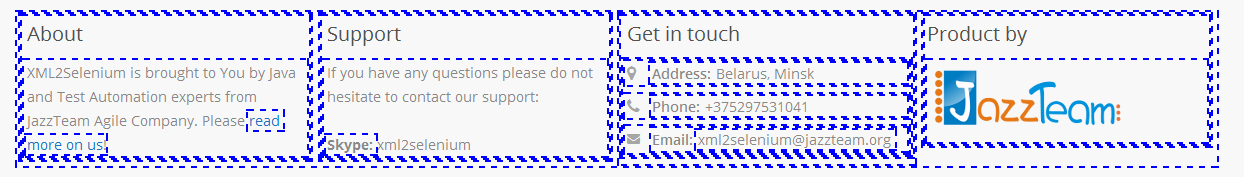
2.1 Выбрать все элементы (<header>);

2.2 Вернуть все элементы (<div>) которые находятся на второй позиции, на том же уровне что и элемент (<header>);



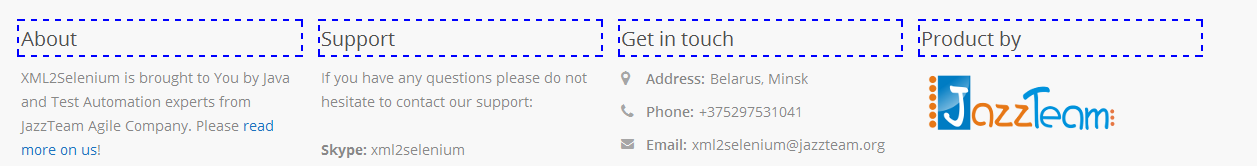
2.3 Сделать рекурсивный спуск;

2.4 Вернуть множество элементов на том же уровне что и (<div>);



2.5 Сделать рекурсивный спуск;

2.6 Выбрать все элементы (<h3>).



[ответ](#id.kquj905ohyn)

3. Придумать 3 XPath самостоятельно.

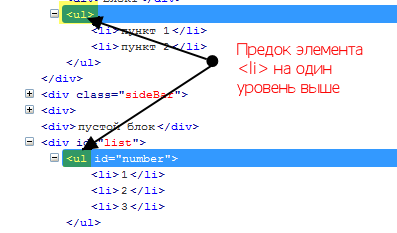
/>

### parent

**parent::** — можно заменить на **«..»** — возвращает предка на один уровень назад.

Пример:

//div/ul/li/parent::ul вернет предка элемента li, можно заменить на //div/ul/li/..



/>

Задание: Откроем страницу, для которой мы пишем примеры (см. [Необходимые инструменты](#h.fli2cx14v1wf)).

1.1 Выбрать все элементы (<p>);

1.2 Вернуться на один уровень выше, к элементу (<div>).

[ответ](#id.5wq7w88gct1m)

2.1 Выбрать все элементы (<a>);

2.2 Подняться на один уровень выше к элементу (<li>) у которого есть атрибут @id;

2.3 Подняться на один уровень выше;

2.4 Подняться на один уровень выше к элементу (<li>) у которого есть элемент (<a>) с классом @class="sf-with-ul".

[ответ](#id.5wq7w88gct1m)

3. Придумать 3 XPath самостоятельно.

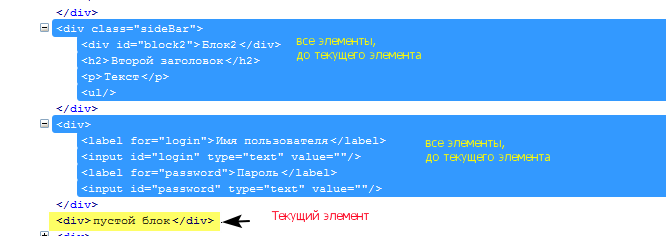
/>

### 

### preceding

**preceding::** — возвращает множество обработанных элементов включая множество предков.

Пример: //div[[text()](#id.meq8sgz85u3u)='пустой блок']/preceding::\* обработает все элементы, до элемента //div[text()='пустой блок']



/>

Задание: Откроем страницу, для которой мы пишем примеры (см. [Необходимые инструменты](#h.fli2cx14v1wf)).

1.1 Вернуть элемент (<div>) с атрибутом @id="footer-bottom";

1.2 Вернуть все необработанные элементы выше текущего.

[ответ](#id.v96tn8vaqr6d)

2.1 Вернуть элемент (<div>) с атрибутом @id="footer-bottom".;

2.2 Вернуть все необработанные элементы (<div>) у которого есть элемент (<p>) с потомком (<img>);

2.3 Вернуть элемент (<a>).

[ответ](#id.v96tn8vaqr6d)

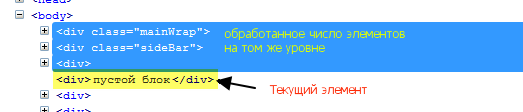
3. Придумать 3 XPath самостоятельно.

### 

### preceding-sibling

**preceding-sibling::** — возвращает множество элементов на том же уровне, предшествующих текущему.

Пример: //div[text()='пустой блок']/preceding-sibling::\* вернет обработанное число элементов на том же уровне(братья и сестры), что и элемент //div[text()='пустой блок'], но сам элемент //div[text()='пустой блок'] не обработается



/>

Задание: Откроем страницу, для которой мы пишем примеры (см. [Необходимые инструменты](#h.fli2cx14v1wf)).

1.1 Выбрать все элементы (<div>) с атрибутом @id="tab4";

1.2 Вернуть множество элементов (<div>) у которого есть элемент (<p>) с потомком (<a>);

1.3 Сделать рекурсивный спуск;

1.4 Вернуть все элементы (<a>) которые находятся на первой позиции.

[ответ](#id.3xgyok5hm274)

2.1 Выбрать все элементы (<selection>) с атрибутом @class="comments-sec";

2.2 Вернуть множество элементов (<div>) у которого есть элемент (<h2>);

2.3 Вернуть элемент (<div>);

2.4 Вернуть элемент (<div>);

2.5 Вернуть множество элементов (<div>) у которого есть атрибут @class="columns sixteen".

[ответ](#id.3xgyok5hm274)

3. Придумать 3 XPath самостоятельно.

### 

### self

**self::** можно заменить на **«.»** — возвращает текущий элемент. //self::p

Задание: Откроем страницу, для которой мы пишем примеры (см. [Необходимые инструменты](#h.fli2cx14v1wf)).

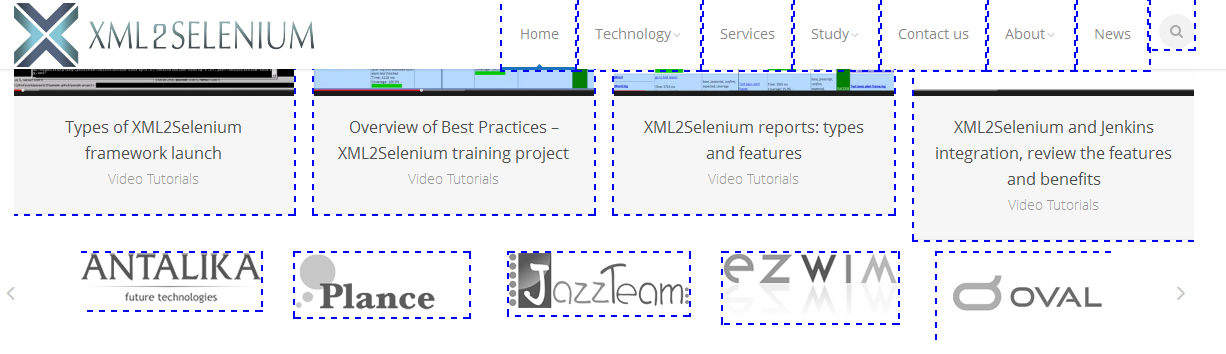
1. Составить 5 XPath самостоятельно.

/>

**Итоговое задание:**

Откроем страницу, для которой мы пишем примеры (см. [Необходимые инструменты](#h.fli2cx14v1wf)).

1.1 Выбрать все элементы (<li>);



1.2 Выбрать всех потомком (<img>) - не сокращенный синтаксис;



1.3 Подняться на один уровень выше;

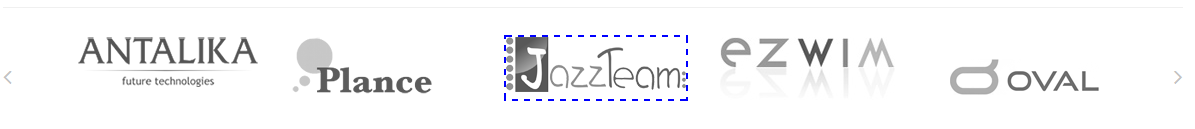
1.4 Подняться на один уровень выше - не сокращенный синтаксис;

1.5 Подняться на один уровень выше - не сокращенный синтаксис;



1.6 Сделать рекурсивный спуск к элементу (<li>) который находится

на третьей позиции;



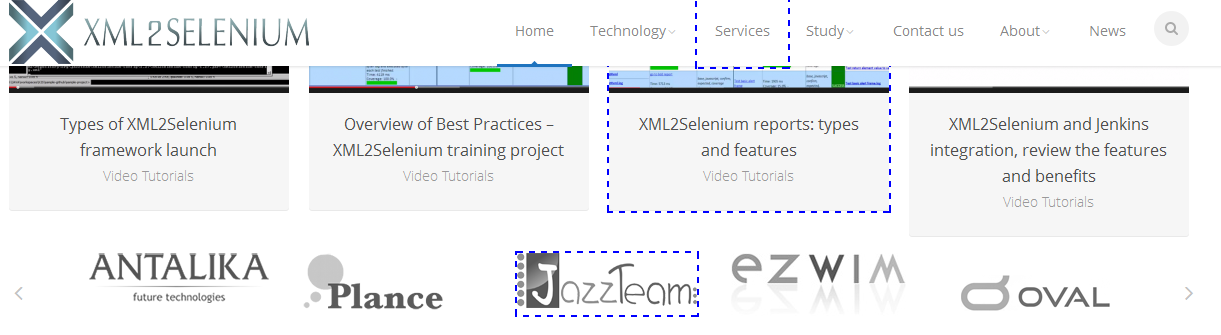
1.7 Выбрать всех предков ;



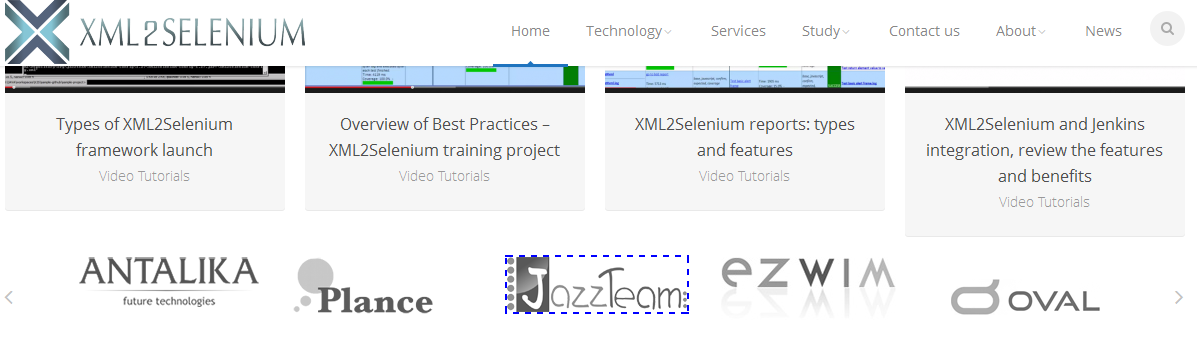
1.8 Сделать рекурсивный спуск к элементу (<div>) в котором есть элемент (<li>) c потомком (<img>);



1.8 Сделать рекурсивный спуск к элементу (<li>) который находится на третьей позиции;



1.9 Выбрать потомка (<img>) элемента (<li>).



[ответ](#id.4r7dfrbhx3md)

/>

Пример использования XPath на эту тему:

//div[@class='b-lineb-line\_lastl-widgets']//tr[@class='widgets-line']/td[@style]//parent::div//div/attribute::class/../child::\*/attribute::href/parent::\*/parent::\*/descendant::span/span[@class='b-region\_\_cityname\_text']

# 

# Функции XPath

XPath содержит библиотеку встроенных функций для преобразования данных.

## Функции обработки наборов узлов

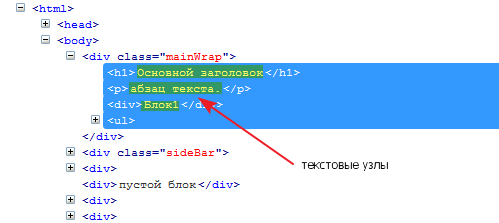
Функции набора узлов обеспечивают информацию о наборе узлов (одном узле или более). В число функций набора узлов входят:

### node()

*node-set* **node()** - возвращает все узлы. В отличие от «\*» функция возвращает также текстовые узлы.

Пример:

//div[@class="mainWrap"]/node()



/>

Задание: Откроем страницу, для которой мы пишем примеры (см. [Необходимые инструменты](#h.fli2cx14v1wf)).

1. Создать 5 XPath в котором будет присутствовать функция node().

/>

Пример использования XPath на эту тему: //div/node()//a/node()/span

### 

### text()

*String* **text()** - возвращает набор текстовых узлов. //div[text()='Блок1'] вернет блок div в котором содержится текст “Блок1”

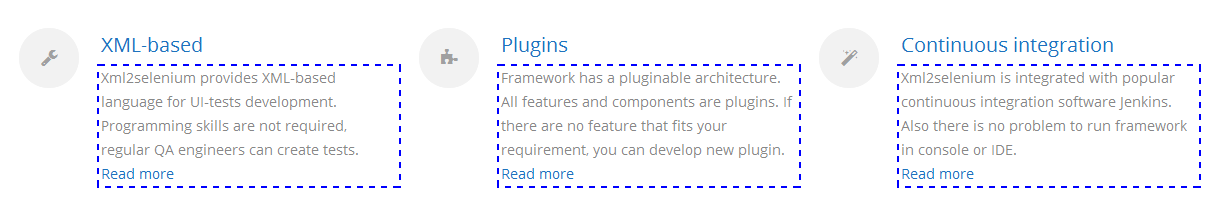
Задание: Откроем страницу, для которой мы пишем примеры (см. [Необходимые инструменты](#h.fli2cx14v1wf)).

1. Выбрать все элементы в которых содержится текст "Read more".



[ответ](#id.gnz5m74i0hen)

2. Выбрать все элементы (<p>) в которых есть элемент (<a>) с текстом "Read more".



[ответ](#id.gnz5m74i0hen)

3. Выбрать весь текст, который содержится в элементах (<div>).

[ответ](#id.gnz5m74i0hen)

4. Придумать 3 XPath самостоятельно.

/>

Пример использования XPath на эту тему: //div[text()]//span[text()='Весенние фоны']

### 

### position()

*number* **position()** - возвращает позицию элемента в множестве.

//div[position()=2] -возвращает элементы (<div>) которые находятся на второй позиции. Данный XPath равен //div[2]

Задание: Откроем страницу, для которой мы пишем примеры (см. [Необходимые инструменты](#h.fli2cx14v1wf)).

1. Выбрать все элементы которые находятся на второй позиции.

[ответ](#id.lazmu1mvn8sv)

2. Выбрать все элементы (<div>) у которых есть любые элементы на 5-ой позиции

[ответ](#id.lazmu1mvn8sv)

3.1 Выбрать все элементы (<div>) которые находятся на 3-ей позиции;

3.2 Сделать рекурсивный спуск;

3.3 Выбрать все элементы (<div>) которые стоят на 2 второй позиции и выше, и содержат атрибут @class;

3.4 Сделать рекурсивный спуск.;

3.5 Выбрать ввесь текст;

3.6 Подняться на один уровень выше.

[ответ](#id.lazmu1mvn8sv)

4. Придумать 3 XPath самостоятельно.

/>

Пример использования XPath на эту тему: //div[position()=2]/div[position()<2]/span[position()>3], как говорилось [выше](#id.yra5o5z4qoxk), в [] можно использовать операции отношений

### 

### last()

*number* **last()** — возвращает номер последнего элемента в множестве. Функция first() не предусмотрена. Для доступа к первому элементу используйте индекс «1» //div[1].

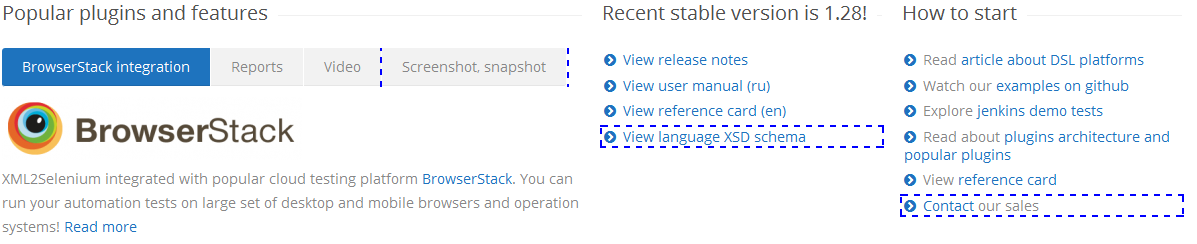
Пример:

//div/label[last()] - вернет последний элемент (<label>), которые содержатся в элементе (<div>)

/>

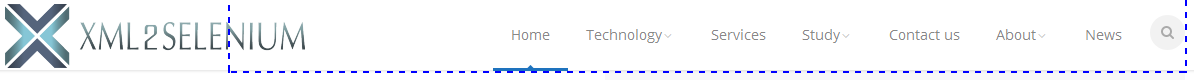
Задание: Откроем страницу, для которой мы пишем примеры (см. [Необходимые инструменты](#h.fli2cx14v1wf)).

1 Вернуть все элементы (<li>) которые находятся на последней позиции.



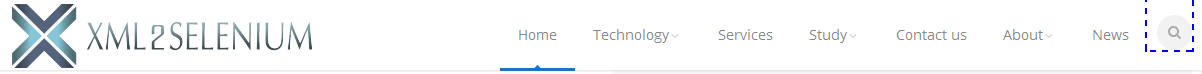
[ответ](#id.c7ty60n8jd9k)

2.1 Вернуть все элементы (<div>) в которых есть элементы (<nav>);



2.2 Сделать рекурсивный спуск;

2.3 Выбрать элемент (<li>) котоый находиться на последней позиции.



[ответ](#id.c7ty60n8jd9k)

3. Придумать 3 XPath самостоятельно.

/>

Пример использования XPath на эту тему: //div[last()]//div[last()]/p[last()]

### 

### count()

*number* **count(node-set)** — возвращает количество элементов в node-set.

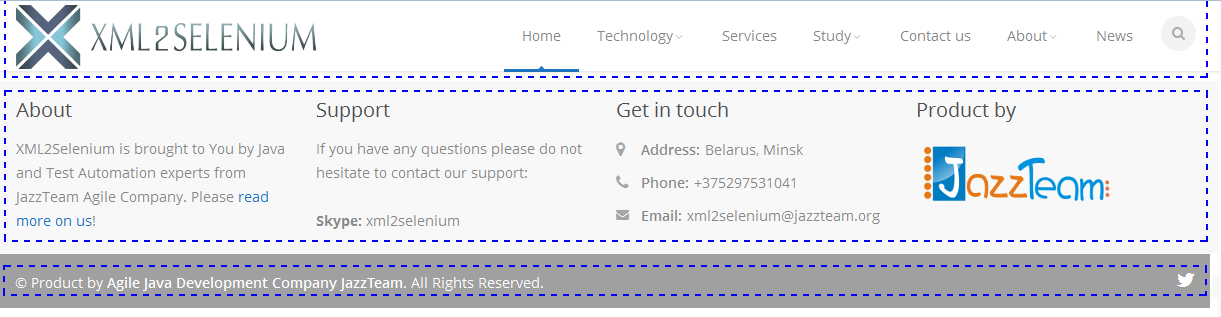
//div[count(input)=2] вернет блок div, в котором содержится 2 элемента input

Задание: Откроем страницу, для которой мы пишем примеры (см. [Необходимые инструменты](#h.fli2cx14v1wf)).

1. Выбрать все элементы (<div>) в которых содержится ровно 3 элемента (<div>)

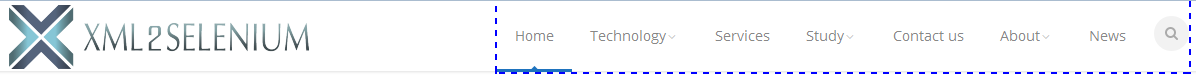
[ответ](#id.zibejvigzn5d)

2.1 Выбрать все элементы (<div>) в которых есть атрибут @class="container";



2.2 Сделать рекурсивный спуск;

2.3 Выбрать все элементы в которых количество любых элементов равно 8.



[ответ](#id.zibejvigzn5d)

3. Придумать 3 XPath самостоятельно.

/>

Пример использования XPath на эту тему: //div[count(div)=2]//div[count(span)>1]/a[count(p)=1]

### 

### id()

*node-set* **id(object)** - находит элемент с уникальным идентификатором.

id('login') - вернет элемент в котором присутствует атрибут id со значением login

Задание: Откроем страницу, для которой мы пишем примеры (см. [Необходимые инструменты](#h.fli2cx14v1wf)).

1. Выбрать элементы с id=”footer”, id=”logo”.

/>

Пример использования XPath на эту тему: id('login')

## 

## Строковые функции

### string()

*string* **string(object)** — возвращает текстовое содержимое элемента. По сути возвращает объединенное множество текстовых узлов на один уровень ниже.

string(//ul[@id="number"]) вернет все текстовое содержимое элемента (<ul[@id="number"]>) результат выполнения функции 1 2 3

Задание: Откроем страницу, для которой мы пишем примеры (см. [Необходимые инструменты](#h.fli2cx14v1wf)).

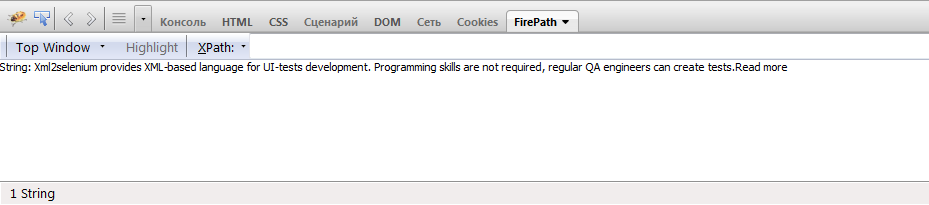
1.1 Выбрать элемент (<div>) в котором содержится @class='one-third alpha columns';

1.2 Сделать рекурсивный спуск;

1.3 Выбрать элемент (<p>).

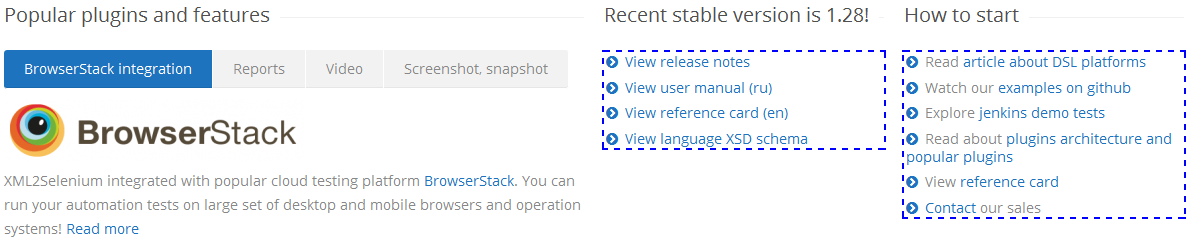


1.4 Вернуть текстовое содержимое элемента (<p>) .



[ответ](#id.n43k77safahu)

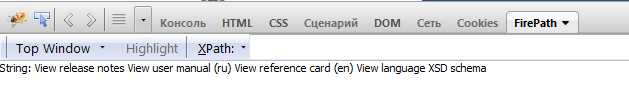
2.1 Выбрать все элементы (<div>) в которых присутствует атрибут @class='list-4';



2.2 Выбрать все элементы (<ul>) в которых количество элементов (<li>) равно 4;



2.3 Вернуть весь текст элемента (<ul>).



[ответ](#id.n43k77safahu)

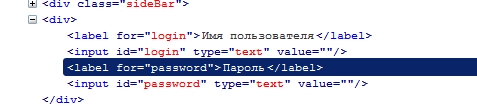
3. Придумать 3 XPath самостоятельно.

### 

### contains()

*boolean* **contains(string, string)** - возвращает true, если первая строка содержит вторую, иначе возвращает false.

Пример: //\*[contains(text(), 'Пароль')] вернет все элементы, которые содержат текст “Пароль”



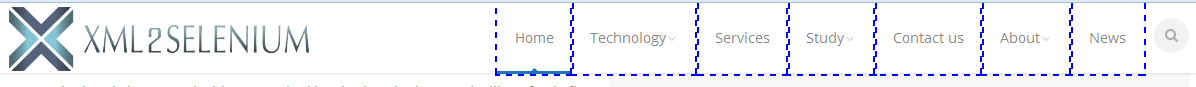
/>

Задание: Откроем страницу, для которой мы пишем примеры (см. [Необходимые инструменты](#h.fli2cx14v1wf)).

1. Выбрать все элементы (<h3>) в которых содержится текст 'How to start'.

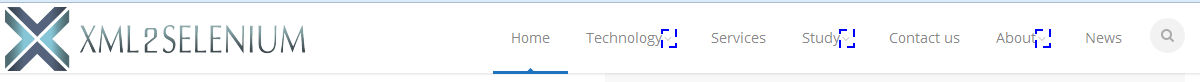
[ответ](#id.c5z7bt3isatq)

2.1 Выбрать все элементы (<li>) в атрибуте @id которых содержится текст 'menu-item';



2.2 Сделать рекурсивный спуск;

2.3 Выбрать все элементы (<i>).



[ответ](#id.c5z7bt3isatq)

3.1 Выбрать все элементы (<div>) в атрибуте @class которых содержится текст 'featured';

3.2 Выбрать элемент (<h3>);

3.3 Выбрать элемент (<a>) в котором содержится текст 'Plugins'.

[ответ](#id.c5z7bt3isatq)

4. Придумать 3 XPath самостоятельно.

### 

### concat()

*string* **concat(string, string, string\*)** - объединяет две или более строк

Пример: //li[contains(text(), concat(1, 2, 3))]

данных XPath объединит строки “1” ”2” ”3” и произведет поиск по элементу (<li>) в котором содержится текст “123”

/>

Задание: Откроем страницу, для которой мы пишем примеры (см. [Необходимые инструменты](#h.fli2cx14v1wf)).

1. Зыбрать элемент (<a>) в котором содержится текст объедененных 3-х строк

'Continuous integration'.

[ответ](#id.tvfjvu30dxx2)

2. Выбрать элемент (<li>) в которых содержится атрибут @id с текстом из объедененных 3-х строк 'menu-item-1478' .

[ответ](#id.tvfjvu30dxx2)

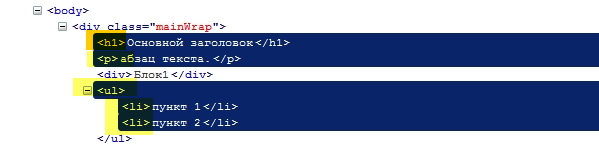
3. Придумать 3 XPath самостоятельно.

### 

### string-length()

*number* **string-length(string)** - возвращает длину строки.

Пример: //\*[string-length(name())<=2] вернет все элементы, длинна имени тегов которых меньше либо равно 2. //\*[string-length(text())>10] вернет все элементы, длинна текста которых больше 10 символов



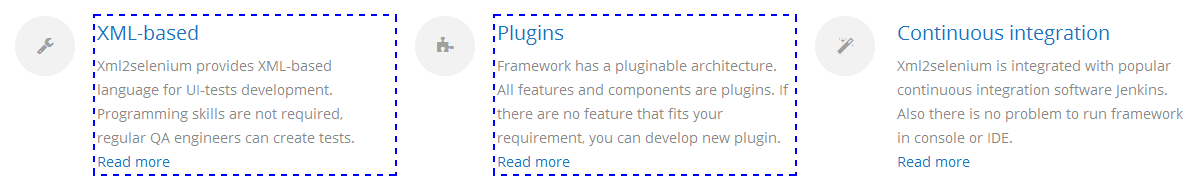
/>

Задание: Откроем страницу, для которой мы пишем примеры (см. [Необходимые инструменты](#h.fli2cx14v1wf)).

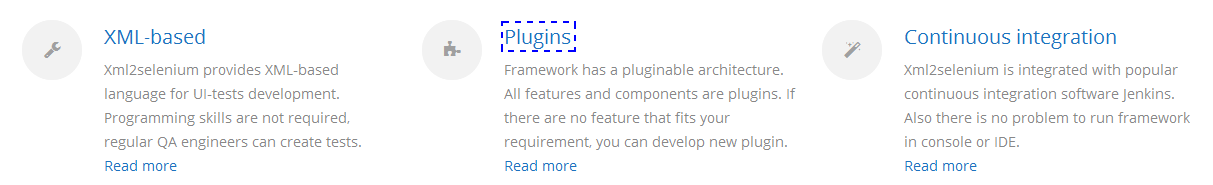
1. Выбрать все элементы у которых длинна текста, который содержится в элементе (<p>) больше 150 символов.

[ответ](#id.th774qirdbh7)

2.1 Выбрать все элементы у которых длинна текста, который содержится в элементе (<p>) больше 150 символов;



2.2 Выбрать все элементы (<a>) длинна текста в которых равна 7.



[ответ](#id.th774qirdbh7)

3. Придумать 3 XPath самостоятельно.

/>

### substring()

*string* **substring(‘string’, number, number)** - возвращает строку вырезанную из строки начиная с указанного номера, второй параметр number (необязательный) — количество символов.

Пример: //\*[contains(text(), substring('Блок1вавав', 1, 5))]

1.substring('Блок1вавав', 1, 5) функция возьмет 5 символов, начиная с первого. Результат выполнения функции “Блок1”.

2.далее в XPath подставиться результат выполнения функции substring

//\*[contains(text(), “Результат выполнения функции substring”].

3. теперь XPath будет искать на странице элементы в которых содержится текст “Блок1”.

/>

Задание: Откроем страницу, для которой мы пишем примеры (см. [Необходимые инструменты](#h.fli2cx14v1wf)).

1.1 Выбрать все элементы в которых содержится атрибут @id с текстом ‘aq-block-880-2’;

1.2 Текст 'aq-block-880-2' обрезать 6 символов, начиная с 8 символа .

[ответ](#id.2j5pur3h73r1)

2. Придумать 3 XPath самостоятельно.

/>

### substring-before()

*string* **substring-before(string, string)** - если найдена вторая строка в первой, возвращает строку до первого вхождения второй строки.

Пример: //\*[contains(text(), substring-before('Блок1/блабла/', '/'))]

1. substring-before('Блок1/блабла/', '/') обрежет весь текст, который встречается после первого “/” включая сам “/”. Результат “Блок1”.

2. далее в XPath подставиться результат выполнения функции substring-before.

3. теперь XPath будет искать на странице элементы в которых содержится текст “Блок1”. //\*[contains(text(), Блок1)]

/>

### substring-after()

*string* **substring-after(string, string)** - если найдена вторая строка в первой, возвращает строку после первого вхождения второй строки.

//\*[contains(text(), substring-before('блабла/Блок1', '/'))] все тоже самое, что в предыдущей функции, только теперь обрежется текст до “/” включая сам “/”.

### 

### starts-with()

*boolean* **starts-with(string, string)** - возвращает true если первая строка начинается со второй, иначе возвращает false.

Пример: //h1[starts-with('div', 'd')]

функция starts-with('div', 'd') проверит, присутствуют ли на странице элементы “div” которые начинаются с “d”

/>

Задание: Откроем страницу, для которой мы пишем примеры (см. [Необходимые инструменты](#h.fli2cx14v1wf)).

1. Выбрать все элементы (<a>) которые начинаются с текста 'View'

[ответ](#id.x6rp8yy3vg95)

2. Придумать 3 XPath самостоятельно.

/>

### 

### normalize-space()

*string* **normalize-space(string)** - убирает лишние и повторные пробелы, а также управляющие символы, заменяя их пробелами.

Пример: //\*[contains(text(), normalize-space(' пункт 1 '))]

normalize-space(' пункт 1 ') обрежет лишние пробелы и вернет текст в нормально виде “пункт 1”.

/>

### 

### translate()

*string* **translate(string, string, string)** - заменяет символы первой строки, которые встречаются во второй строке, на соответствующие позиции символам из второй строки символы из третьей строки. translate(«cat», «abc», «ABC») вернет CAt. //\*[contains(text(), translate('БЛОК1', 'ЛОК', 'лок'))], вернет “Блок1”.

## 

## Логические функции

### логическое «или»

**or** — логическое «или»

Пример:

//\*[@class="mainWrap" or @class="tr1"] данный XPath выберет все элементы, которые указаны в предикате, если элемента с классом class="mainWrap" нету, то выберутся все элемменты с классом class="tr1" и на оборот. Можно указывать 1 или больше *or*

//\*[@class="mainWrap" or @class="tr1" or @border="1" or @id="block2"]

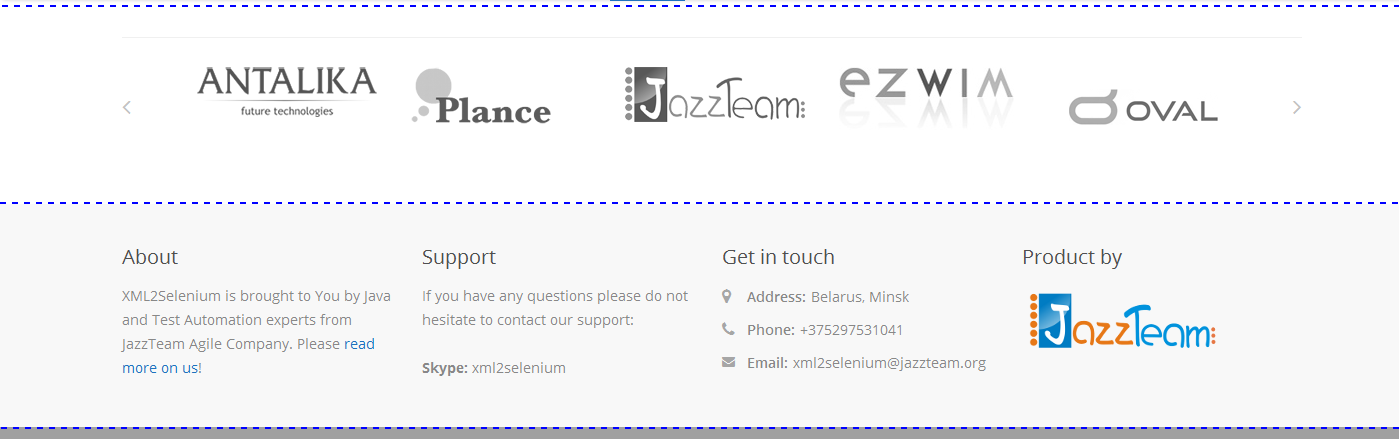
/>

Задание: Откроем страницу, для которой мы пишем примеры (см. [Необходимые инструменты](#h.fli2cx14v1wf)).

1. Выбрать все элементы (<div>) в которых содержаться атрибуты @id или @class

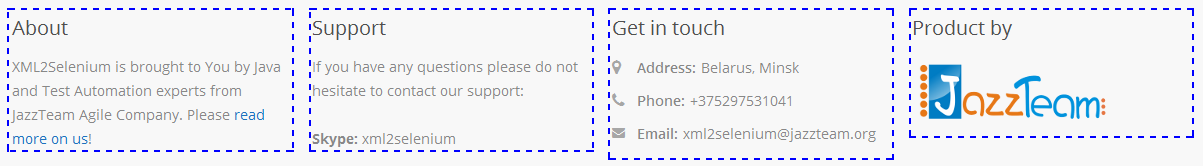
[ответ](#id.8g44mwr7snwb)

2.1 Выбрать все элементы в которых содержится атрибут @id='header' или @id='footer’;

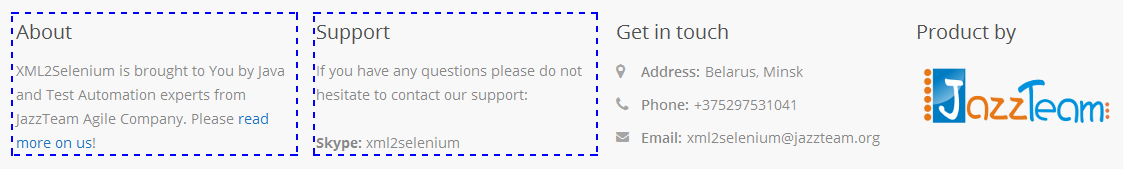


2.2 Сделать рекурсивный спуск до элемента (<div>) в котором содержатся атрибуты

@class='four columns' или @class='columns';



2.3 Выбрать все элементы у которых есть атрибут @id='text-2' или @id='text-7'.



[ответ](#id.8g44mwr7snwb)

3. Придумать 3 XPath самостоятельно.

/>

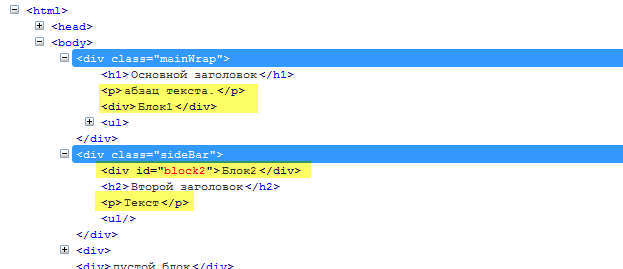
### 

### логическое «и»

**and** — логическое «и»

Пример:

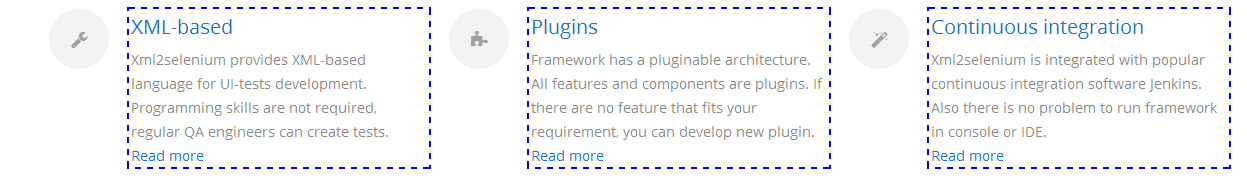
//div[div and p] выражение находит все элементы (<div>), содержащие хотя бы по одному элементу (<div>) и (<p>).



/>

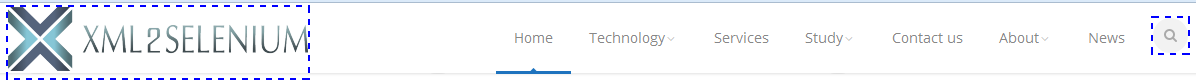
Задание: Откроем страницу, для которой мы пишем примеры (см. [Необходимые инструменты](#h.fli2cx14v1wf)).

1. Выбрать элементы (<div>) в которых есть элементы (<h3>) и (<p>).



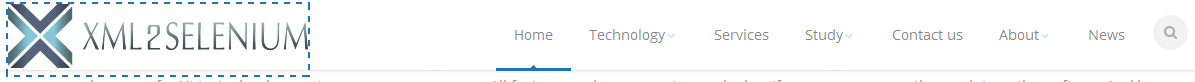
[ответ](#id.2sfpfw1t59er)

2ж1 Выбрать все элементы (<div>) в которых содержатся атрибуты @id и @style.



2.2 Сделать рекурсивный спуск до элемента (<a>) в котором содержатся атрибуты

@rel='home' or @titleю



[ответ](#id.2sfpfw1t59er)

3. Придумать 3 XPath самостоятельно.

/>

### логическое «равно»

**=** — логическое «равно»

Пример: //\*[string-length(name())=5] Вернет все элементы, длинна имени которых равна 5.

### 

### логическое «меньше»

**< (<)** — логическое «меньше»

Пример: //\*[position()<3] вернет все элементы, которые находятся на позиции не больше 3.

### 

### логическое «больше»

**> (>)** — логическое «больше»

Пример: //div[count(input)>1] вернет все элементы (<div>), в которых содержится больше чем один элемент (<input>).

### 

### логическое «больше либо равно»

**= (>=)** — логическое «больше либо равно»

Пример: //\*[position()>=3] вернет все элементы, которые находятся на позиции равной 3 или больше.

## Булевы Функции

### 

### boolean()

*boolean* **boolean(object)** - приводит объект к логическому типу;

Пример: boolean(//input[@id="login"]) вернет, true, т.к. на странице присутствует элемент по указанному адресу.

### 

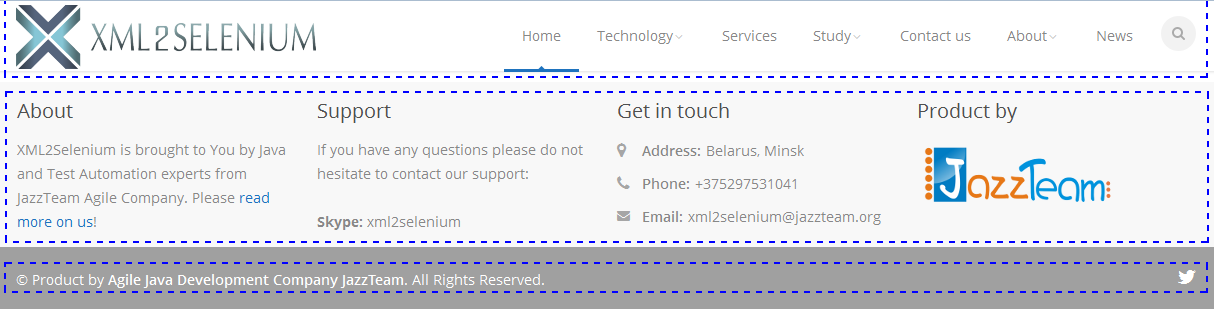
### true()

*oolean* **true()** -возвращает истину.

Пример: //input[@id="login"=true()] данный XPath идентичен XPath //input[@id="login"]

Задание: Откроем страницу, для которой мы пишем примеры (см. [Необходимые инструменты](#h.fli2cx14v1wf)).

1.1 Выбрать все элементы (<div>) в которых содержится атрибут @class="container";



1.2 Выбрать все элементы (<div>) в котором @class="eight columns" возвращает истину;

Image 1.png

1.3 Cделать рекурсивный спуск и выбрать все элементы (<i>).

Image 1.png

[ответ](#id.uslwqni64tdz)

2. придумать 3 XPath самостоятельно

/>

### 

### false()

*boolean* **false()** - возвращает ложь.

Пример: //input[@id="password"=false()] данный XPath найдет все элементы (<input>), и вернет первый найденный элемент из списка, т.к. элемент с ([@id="password"]) установлен как false

Задание: Откроем страницу, для которой мы пишем примеры (см. [Необходимые инструменты](#h.fli2cx14v1wf)).

1.1 Выбрать все элементы у которых есть атрибут @id='header';

1.2 Сделать рекурсивный спуск к элементу (<li>);



1.3 Выбрать все элменты (<a>) кроме тех в которых присутствует текст "S".



[ответ](#id.yucl18fhckue)

2. придумать 3 XPath самостоятельно

/>

### 

### not()

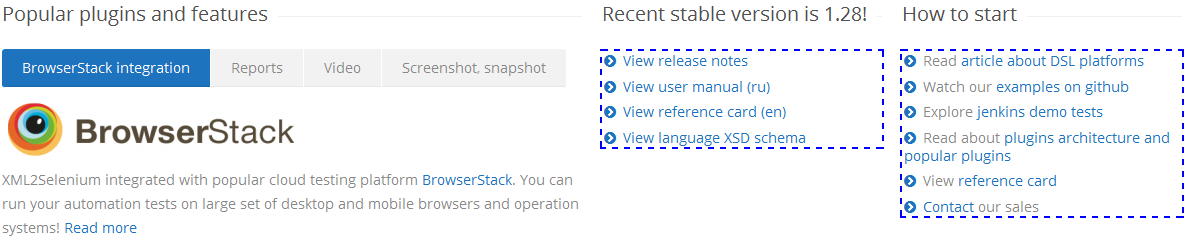
*boolean* **not(boolean)** - логическое отрицание, возвращает true если аргумент false и наоборот.

пример: //input[not(@id="password"=false())]

Задание: Откроем страницу, для которой мы пишем примеры (см. [Необходимые инструменты](#h.fli2cx14v1wf)).

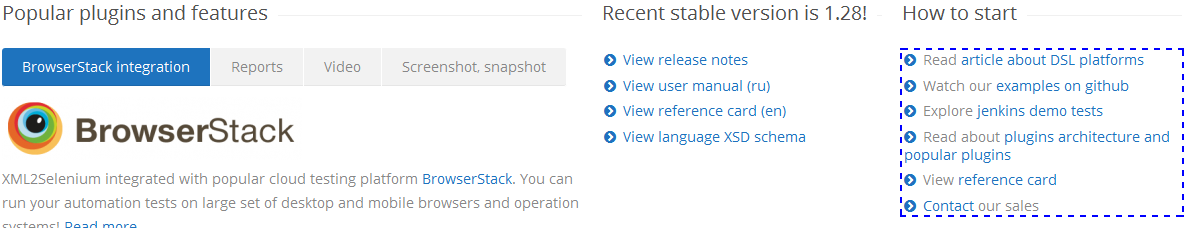
1.1 Выбрать все элементы у которых присутствует атрибут @class="list-4" ;

1.2 Создать массив из полученных элементов ();



1.3 Элемент, который находиться на второй позиции сделать false;

1.4 Элемент, который находится на второй позиции сделать true.



[ответ](#h.xj46pnyzt0vs)

2. придумать 3 XPath самостоятельно

/>

## 

## Числовые функции

### 

### Cложение

**сложение(+)**

Пример: //div[1+3] вернет 4-й элемент (<div>) из коллекции.

### 

### Вычитание

**вычитание(-)**

Пример: //\*[string-length(name())=4-1] вернет все элементы у которых длинна имени 3 символа.

### Умножение

**умножение(\*)**

Пример: //\*[position()=2\*2] вернет все элементы, которые находятся на позиции 4 .

### div

**div** — обычное деление (не деление нацело!)

Пример: //\*[position()=3 div 2] не вернет ни одного элемента, т.к. нету элемента, который находиться на позиции 1.5.

### mod

**mod** — остаток от деления

Пример: //\*[position()=5 mod 3] вернет все элементы, которые находятся на 2 позиции.

### number()

*number* **number(object)** - переводит объект в число.

Пример: number(//ul[@id='longNumber']/li[2][text()]) данный XPath преобразует текст полученный из (li[2]) в число. Теперь с этим числом можно производить разные математические действия.

### sum()

*number* **sum(node-set)** - возвращает сумму множества, каждый тег множества

будет преобразован в строку и из него получено число.

Пример: sum(//ul[@id='number']/li) вернет число 6, т.к. в каждом (<li>) элементе находится число. Функция сложит все числа изо всех элементов.

li[1] = 1

li[2] = 2

li[3] = 3

1+2+3 = 6

/>

### 

### floor()

*number* **floor(number)** - округление аргумента в меньшую сторону.

Пример: //\*[floor(5 div 2)] вернет все элементы, которые находится на позиции 2.

### 

### ceiling()

*number* **ceiling(number)** — округление элемента в большую сторону.

Пример: //\*[floor(5 div 2)] вернет все элементы, которые находится на позиции 3.

### 

### round()

*number* **round(number)** - округление числа по математическим правилам.

Пример: //\*[round(2.5)] вернет все элементы, которые находятся на 3-й позиции. Число округлится до 3 .

//\*[round(2.4)] вернет все элементы, которые находятся на 2-й позиции.

Число округлится до 2.

### 

# Советы по написанию хорошего XPath

### 1. ***Постарайтесь избегать абсолютных путей в XPath запросах***.

Если вы используете какие либо инструменты (fireBug) для получения XPath-путей элементов, это совсем не значит, что возвращаемый ими XPath единственно правильный. Чаще такие инструменты возвращает как раз таки абсолютные пути.

**Примеры Xpath, которые делать не надо**:

/html/body/div/div/div/div/ul/li/div/a[@class=”class”]

Дело в том, что такой XPath, тяжело поддерживать в актуальном состоянии. Представьте себе, что разработчик добавил на страницу новые блоки (<div>)

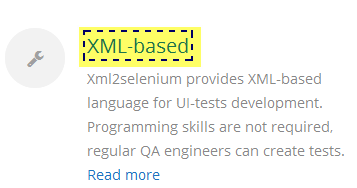
/html/body/div/div/div/div/div/div/ul/li/div/a[@class=’class’]

Нам придется постоянно менять XPath, что займет определенной количество времени. Т.к. мы используем блок div в котором содержится тег a с атрибутом class, то лучше написать такой XPath

//div/a[@class=”class”]

### 2. ***Старайтесь не использовать цифры.***

Откроем страницу, для которой мы пишем примеры (см. [Необходимые инструменты](#h.fli2cx14v1wf)). Если воспользоваться firebug, и нажать на этот элемент



то нам выдаст следующий XPath

.//\*[@id='aq-block-880-2']/div[1]/div/div[2]/h3/a

Опять же как и в первом пункте это не очень удобно, из-за того, что страничка может быть динамичной, и могут добавляться новые блоки (<div>). Вместо цифр, лучше использовать атрибуты

.//\*[@id='aq-block-880-2']/div[@class="one-third alpha columns"]/div/div[@class="featured-desc"]/h3/a

### 3. ***Не использовать длинные*** *XPath****.***

.//\*[@id='aq-block-880-2']/div[@class="one-third alpha columns"]/div/div[@class="featured-desc"]/h3/a

Данный XPath правильный , но он очень длинный. Поддерживать такой XPath в актуально состоянии сложнее. Сократим XPath

//div[@class="one-third alpha columns"]//h3/a

Сокращенный XPath более читабельный, и работать с ним будет намного легче.

### 4. ***Cтарайтесь не использовать \* перед квадратными скобками***.

*(только если в этом нету необходимости)*

Например: у нас есть след XPath //\*[@class="featured-desc"]/h3/a но на странице присутствует тег (<span>) и тег (<div>) с одинаковой вложенностью элементов и одинаковым атрибутом класс, и данный XPath будет не уникальным.

Для того что бы получить уникальный XPath, нам придется добавлять новые узлы, что приведет к увеличению XPath. Указывайте явно, с каким тегом работаете

//div[@class="featured-desc"]/h3/a

/>

### 5. ***Cтарайтесь не использовать подзапросы.***

Чаще всего подзапросы появляются в XPath-запросе при наличии идентификаторов у соседних или дочерних элементов, и их отсутствии у искомого элемента.

Например, нужно получить элемент ссылки (<a>):

<div class=”div”>

<div id="div1-id">некоторый div</div>

<div>еще один div</div>

<a href="<http://internetka.in.ua/>">

<img src="url" alt="картинка-ссылка" />

</a>

</div>

<div class=”div”>

<div id="div2-id">некоторый div</div>

<div>еще один div</div>

<a href="<http://internetka.in.ua/>">

<img src="url" alt="картинка-ссылка" />

</a>

</div>

Первое, что бросается в глаза — это идентификаторы ([@id="div1-id"]) и ([@id="div2-id"]). Будет вполне нормально использовать один из них для поиска ссылки.

//div[div[@id='div1-id']]//a

этот запрос будет правильным, но работать он будет медленнее, потому что ([@id="div1-id"]) будет выполняться для всех найденных (<div>) на странице.

Конечно, использование подзапросов в XPath в некоторых ситуациях может быть действительно эффективно. Но сравним его с другим вариантом выполнения той же задачи:

//div[“id="div1-id"]/..//a

В данном XPath осуществляется поиск по id элемента и от него строится остальной путь. В среднем последний пример будет работать быстрее.

# Примеры сложных XPath

Примеры сложных XPath, будем разбирать на базе исходного кода страницы, о которой упоминается в инструментах в начале данного документа.

**XPath 1**:

//div[contains(h3/text(), 'Testimonials')]//following-sibling::div//a[contains(text(), 'BrowserStack')]

Разберем:

//div[contains(h3/text(), 'Testimonials')]

вернет все элементы (<div>) в который присутствует элемент (<h3>) с текстом “Testimonials”

//following-sibling::div

вернет все элементы (братьев и сестер) которые находятся на том же уровне, что и элемент (<div>)

//a[contains(text(), 'BrowserStack')]

вернет все элементы (<a>) в которых присутствует текст 'BrowserStack'

**XPath 2**:

//div[@class="hideShowTestCaseButtonDiv"]//img[contains(@alt, '-') and contains(@style, 'inline')]

Разберем:

//div[@class="hideShowTestCaseButtonDiv"]

вернет все элементы (<div>) в которых содержится атрибут class со значением

"hideShowTestCaseButtonDiv"

//img[contains(@alt, '-') and contains(@style, 'inline')]

вернет все элементы (<img>) в которых содержатся атрибут alt со значением “-” и атрибут style со значением “display: inline;”

**XPath 3**:

(//a[@class='sf-with-ul' and @href='#'])[1]

Разберем:

(//a[@class='sf-with-ul' and @href='#'])

создаст массив из элементов, которые соответствуют параметрам.

(//a[@class='sf-with-ul' and @href='#'])[1]

выберет первый элемент из созданного массива

**XPath 4**:

//a[contains(text(), 'Read more') and contains(../..//a/text(), 'Plugins')]

Разберем:

данный XPath найдет все элементы (<a>), в которых содержится текст “Read more”, которые находятся в элементе на два уровня ../../ выше, в котором присутствует элемент (<а>) с текстом 'Plugins'

**XPath 5**:

**Пример сложного XPath.**

Данные для этого XPath находятся на [HTML странице](http://xml2selenium.com/xpath/complex/) (страница создана специально для этого XPath).

//td[contains(text(), 'Passat')]/../../../following-sibling::table[position()=1 and count(//a[contains(text(), 'Jazzteam')]) = 1]//tr[not(contains(@class, 'company'))]//a

разберем:

//td[contains(text(), 'Passat')] найдет все (<td>) в которых содержится текст “Passat”

/../../../ поднимемся на 3 уровня выше

following-sibling::table[position()=1 and count(//a[contains(text(), 'Jazzteam')]) = 1] вернем все элементы (<table>) ниже текущего, которые находятся на первой позиции, и в которых присутствует одна ссылка с текстом “Jazzteam”

//tr[not(contains(@class, 'company'))] вернем все классы кроме класса в котором содержится слово “company”

//a выберем ссылку

# Полезные ссылки

[Начало работы с](http://www.ibm.com/developerworks/ru/edu/x-xpath/x-xpath.html) [XPath](http://www.google.com/url?q=http%3A%2F%2Fwww.ibm.com%2Fdeveloperworks%2Fru%2Fedu%2Fx-xpath%2Fx-xpath.html&sa=D&sntz=1&usg=AFQjCNGR1gCjyUHfZ8y9fUntnuKBFug8-Q)

[Справочник по XML-стандартам](http://msdn.microsoft.com/ru-ru/library/ms256177(v=vs.110).aspx)

[XML Path Language (XPath) Version 1.0](http://www.w3.org/TR/xpath)

[Задачи по](http://wiki.jazzteam.org/index.php?title=Xpath/ru) [XPath](http://www.google.com/url?q=http%3A%2F%2Fwiki.jazzteam.org%2Findex.php%3Ftitle%3DXpath%2Fru&sa=D&sntz=1&usg=AFQjCNEVBOPdaGRxZ8PcvsMCI2YJTqxxRQ)

# Ответы на задания

дочерний элемент (/)

1. //div[@id]//h3/a

2. //ul[@id="responsive"]/li/a

3./html/body/div/div/div/div/div/div/div/ul//a

фильтрация ([])

1. //div[3]

2. //div[1][form]

3. //ul[li[5]]//a

4. //div[h3]/p

5. //div[nav][1]//ul[li[6]]//a

ссылка на все элементы (\*)

1. //\*/div[3]

2. //\*[h3]

3. //ul/\*/a

атрибут (@)

1. //div[@class="container"]

2. //div[@id]

3. //div[@class='circle']/i[@class="icon-wrench"]

4. //\*[@\*]//a[@rel]

ссылка на родительский узел (..)

1. //a/..

2. //a/../../div[@id]

3. //h3[/a]/../p/a

группировка операций ()

1. (//li)[10]

общее задание

(//div/div[2][@id][@class]//i/../../div[h3])[2]//p/a[@href]

ancestor

1. //a/ancestor::div[h3]

2. //a/ancestor::div//li/ancestor::ul

ancestor-or-self

1. //ul/li/ancestor-or-self::div

2. //a/ancestor-or-self::ul

3. //span[i]/ancestor-or-self::div[nav]

attribute

1//attribute::class

2//attribute::title

child

1. //div/child::ul

2. //div//child::div[3]

3. //div/child::div[@id]//div[ul]

descendant

1. //div[@id='footer']//div[@class='four columns'][3]/descendant::\*

2. //ul/child::\*

3. //ul/child::li

4. //ul[ancestor::nav][li[6]]/descendant::a

following

1. //div[@id='text-2']/following::\*

2. //div[@id='footer']/following::div[@class='container']/div[1]/following::\*

following-sibling

1. //div[p][h3[@class="headline"]]/h3/following-sibling::p

2. //header/following-sibling::div[2]//following-sibling::\*//h3

parent

1. //p/parent::div

2. //a/parent::li[@id]/parent::\*/parent::li[a[@class="sf-with-ul"]]

preceding

1. //div[@id="footer-bottom"]/preceding::\*

2. //div[@id="footer"]/preceding::div[p/img]//a

preceding-sibling

1. //div[@id="tab4"]/preceding-sibling::div[p/a]//a[1]

2. //section[@class="comments-sec"]/preceding-sibling::div[//h2]/div/div/preceding-sibling::div[@class="columns sixteen"]

не сокращенный синтаксис итоговое задание

//li/child::img/../parent::\*/parent::\*//li[3]/ancestor::\*//div[div//li/img]//li[3]/child::img

text()

1. //\*[text()="Read more"]

2. //p[a[text()="Read more"]]

3. //div/text()

position()

1. //\*[position()=6]

2. //div[\*[5]]

3. //div[position()=3]//div[position()>=2][@class]//text()/..

last()

1. //li[last()]

2. //div[nav]//li[last()]

count()

1. //div[count(div)=3]

2. //div[@class="container"]//\*[count(\*)=8]

string()

1. string(//div[@class='one-third alpha columns']//p)

2. string(//div[@class='list-4']/ul[count(li)=4])

contains()

1. //h3[contains(text(), 'How to start')]

2. //div[contains(@class, 'featured')]/h3/a[contains(text(), 'Plugins')]

concat()

1. //a[contains(text(), concat('Continuous',' ', 'integration'))]

2. //li[contains(@id, concat('menu-','item-', '1478'))]

string-length()

1. //\*[string-length(p[text()])>150]

2. //\*[string-length(p[text()])>150]//a[string-length(text())=7]

substring()

1. //\*[contains(@id, substring('aq-block-880-2', 8, 6))]

strart-with

1. //a[starts-with(text(), 'View')]

логическое «или»

1. //div[@id or @class]

2. //\*[@id='header' or @id='footer']//div[@class='four columns' or @class='columns']/\*[@id='text-2' or @id='text-7']

логическое «и»

1. //div[h3 and p]

true()

//div[@class="container"]/div[@class="eight columns"=true()]//i

false()

//\*[@id='header']//li/a[contains(text(), "S")=false()]

not()

(//div[@class="list-4"])[not(position()=2=false())]