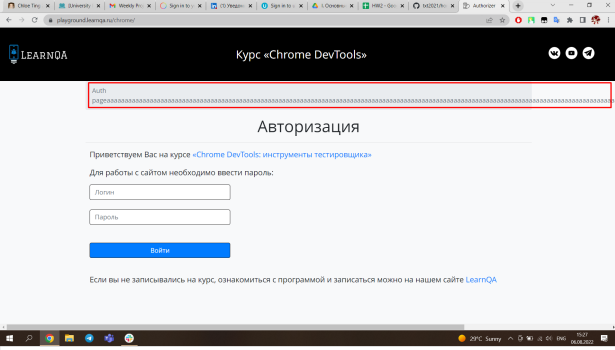
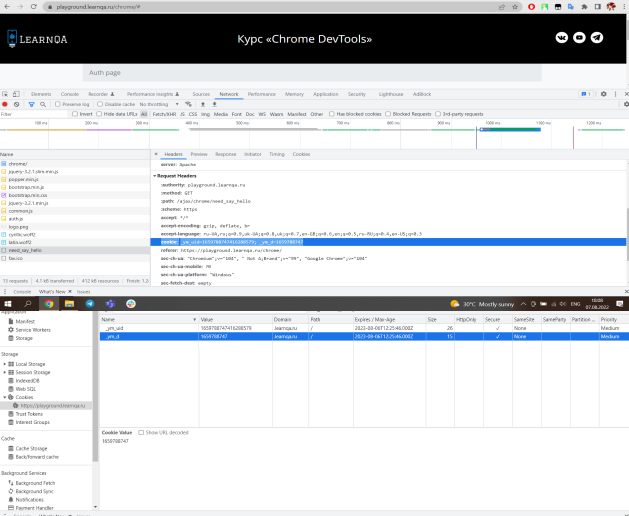
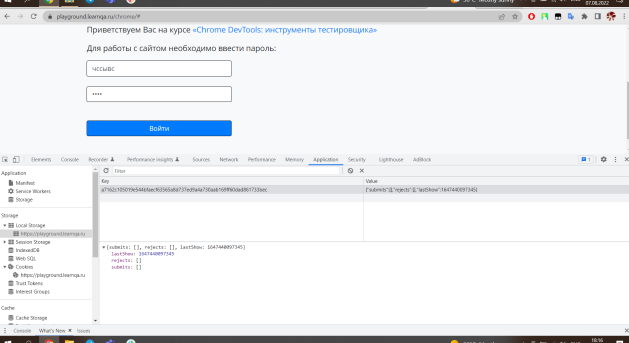


Верно	#	Задача	Решение	Комментарии ученика	Комментарии тренера
		Первый урок			
	Elements	Открываем <a href="http://arsbatyrov.ru/chrome/elements_dz">http://arsbatyrov.ru/chrome/elements_dz</a> Перед нами простой сайт заметок. Наша задача - протестировать его:			
		1. Найти место, где текст выйдет за рамки блока, если будет слишком длинным. <b>Результат:</b> скриншот с вышедшем за рамки текстом.			
		2. Найти и сделать видимым блок нотификации, проверить, адекватно ли он выглядит. <b>Результат:</b> скриншот видимой нотификации. Составить два CSS-локатора: а. Который бы подошел любой кнопке "Детали задания". б. Который бы подошел любой кнопке "Удалить". <b>Результат:</b> написать в решении текстом локаторы.	Authorization is required to enter the page with hw, so it is impossible to find the notification block Due to the reason mentioned above, there is no access to these buttons, so CSS locators were compiled for the "Войти" button - <code>button[class="btn btn-primary"]</code> and the "Логин" input - <code>input[placeholder="Логин"]</code>		
	Console	3. <b>Задача со звездочкой.</b> Составить универсальный XPath-локатор или CSS-локатор, который бы подошел элементам, содержащим текст ТОЛЬКО важных задач (они помечены лейблом "Важное!"). Т.е. в нашем случае элементам, содержащим тексты "Нужно не забыть выполнить домашнее задание." и "Нужно изучить Bash, SQL, Chrome Devtools, ADB, Appium и Selenium, Git". <b>Результат:</b> текстовый файл с локатором.	For the reason mentioned above, the locator that fits all social media icons has been compiled <code>xPath - //*[@id="Layer_1"]</code> CSS -		
		Открываем <a href="http://arsbatyrov.ru/chrome/console_dz">http://arsbatyrov.ru/chrome/console_dz</a> Перед нами галерея с котиками, которые ищут свой дом. Наша задача - протестировать ее:	This link is not available, so the page <a href="https://www.learnqa.ru/">https://www.learnqa.ru/</a> will be tested		
		1. Найти JS-ошибку, которая не блокирует дальнейший функционал. <b>Результат:</b> написать текст ошибки в решении.	Failed to load resource: net::ERR_BLOCKED_BY_CLIENT		
	Console	2. Найти JS-ошибку, которая блокирует функционал закрытия оверлея. <b>Результат:</b> написать текст ошибки в решении.	Can't find blocker on page <a href="https://www.learnqa.ru/">https://www.learnqa.ru/</a>		
		3. Нам надо протестировать кнопку "нажмите сюда" на блоке загрузки. Но этот блок пропадает очень быстро. Надо написать JS-код, который изменит поведение JS-функции, которая уничтожает этот блок. Функция называется иначе, чем в видео, а именно - <code>destroyLoader</code> . <b>Результат:</b> JS-код, который изменит поведение функции + там же ответить на вопрос - верно ли работает кнопка "нажмите сюда" на блоке загрузки или нет. <b>Задача со звездочкой.</b> Для решения этого задания, возможно, понадобится изучение либо урока про вкладку Elements, либо Network, либо Source. JS умеет не только удалять часть HTML-кода, но и добавлять. Так, например, иногда JS дорисовывает нотификацию, а не делает ее просто видимой. Именно такая нотификация у нас есть на странице. Наша задача - протестировать ее. Проблема в том, что мы не знаем ни при каких условиях она показывается, ни название JS-функции, которая ее показывает. Надо изучить JS-код на странице, понять о какой функции идет речь и вызвать ее. <b>Результат:</b> скриншот с показавшейся нотификацией.	<code>destroyLoader = function () {}</code> Can't answer this question because current button is not available		
		Открываем <a href="http://arsbatyrov.ru/chrome/source_dz">http://arsbatyrov.ru/chrome/source_dz</a> Перед нами страница, которая получает картинку со стоков и показывает ее пользователю. Этот сервис нам предстоит протестировать.	Can't do this task because <a href="http://arsbatyrov.ru/chrome/console_dz">http://arsbatyrov.ru/chrome/console_dz</a> page is not available		
		1. Надо посмотреть, с какого стока (или с каких стоков) мы получаем картинки. <b>Результат:</b> написать в решении адресом/адреса.	This link is not available, so the page <a href="https://playground.learnqa.ru/chrome/">https://playground.learnqa.ru/chrome/</a> will be tested <code>playground.learnqa.ru</code> <code>cdnjs.cloudflare.com</code> <code>code.jquery.com</code> <code>maxcdn.bootstrapcdn.com</code>		

Source	<p>2. Надо перегрузить заголовок страницы, чтобы он был не "&lt;title&gt;Source - Home work&lt;/title&gt;", а "&lt;title&gt;Source - локальная версия&lt;/title&gt;". Убедиться, что перезагрузки страницы появляется именно наша локальная версия. <b>Результат:</b> скриншот с заголовком страницы и открытым Chrome DevTools на вкладке Overrides.</p>			
	<p>3. Создать snippet со следующим кодом: "function home_work() {\$( 'a1' ).removeClass( 'a1' ).addClass( 'a2' );home_work();}". Запустить его. После, кликая по кнопке получения новой картинки, понять, что изменилось в поведении кнопки. <b>Результат:</b> написать текстом в решении ответ на вопрос - что изменилось.</p> <p>4. <b>Задача со звездочкой.</b> Наш сервис устроен так, что нужная картинка всегда есть. Наша же задача - протестировать поведение сервиса в случае отсутствия картинки. Делать это будем путем перегрузки JS-файла, который получает путь до картинки и добавляет ее на страницу. Среди загруженных файлов надо найти "source_dz.js". Там есть строка &lt;img class="card-img-top" src="{{path}}" alt="Card image cap"&gt;. Именно вместо {{path}} JS-код подставляет путь до файла. Наша задача поменять эту строку так, чтобы путь до картинки стал неправильным. Далее сохранить изменения в Overrides и перезагрузить страницу. Затем нажать кнопку для получения картинки и посмотреть, как сервис отреагирует. <b>Результат:</b> написать в решении текст с измененной строкой.</p>	<p>Here, the a1 class is removed from all a1's, while a2 class is added</p> <p>&lt;img class="card-img-top" src="C:\Users\Acer\Desktop\testdir" alt="Card image cap"&gt;</p>		
	<p>Открываем <a href="http://arsbatyrov.ru/chrome/network_dz">http://arsbatyrov.ru/chrome/network_dz</a> Перед нами типичный общий чат. Написать сообщение в него может каждый, кто имеет специальную cookie с название chat_token. Наша задача его протестировать:</p> <p>1. Изучить все типы запросов, которые уходят со страницы. Прислать список запросов с описанием того, для чего, по вашему мнению, каждый из них нужен, какой функционал не работал бы без этого запроса. <b>Результат:</b> текст с описанием.</p>	<p>This link is not available, so the page <a href="https://playground.learnqa.ru/chrome/">https://playground.learnqa.ru/chrome/</a> will be tested On this page we can see a different requests, with a script type, stylesheet, document, font and xhr. All of this requests have method GET and 200 or 304 status. With the help of these requests, the relevant data is requested from the server, such as scripts, styles, pictures, fonts.</p>		
	<p><b>Network</b></p> <p>2. Разработчик сказал, что каждый пользователь в чате будет помечен своим цветом. У нас нет других пользователей в чате, но мы можем имитировать другого пользователя при помощи утилиты CUrl. Надо скопировать из вкладки CUrl-запрос на добавление нового сообщения и изменить его так, словно запрос отправил другой пользователь. Затем надо отправить такой запрос и проверить, действительно ли сообщение отобразится новым цветом. <b>Результат:</b> скриншот с чата, где есть сообщения от двух разных пользователей.</p> <p>3. <b>Задача со звездочкой.</b> Найти несколько способов воспользоваться XSS-уязвимостью. Описать каждый из них. <b>Результат:</b> описать в решении с описаниями.</p>	 <p>I don't have access to the chat page, so I just posted a screenshot where I use curl to copy the protocol information</p> <p>Using an XSS vulnerability, an hacker can inject a malicious script into a website that will allow to steal user data, cloud data, payment data etc. This type of vulnerability can be detected both on the client side and on the server side. A malicious script can be injected using various forms on the webpage or through the command line with the help of curl utility.</p>		
Второй урок				
Performance	<p>Открываем <a href="http://arsbatyrov.ru/chrome/performance_dz">http://arsbatyrov.ru/chrome/performance_dz</a> Перед нами такое же JS-приложение, что мы видели на видео. Только кнопки перемешаны. Наша задача - при помощи вкладки Performance понять, какую нагрузку создает каждая из кнопок.</p>			
	<p>1. Надо для каждой кнопки указать тип нагрузки, которые превалирует: "рендеринг и отрисовка", "скриптинг (долгая работа JS, например, для подсчетов чего-либо)", "idle (холостая работа в процессе ожидания)". <b>Результат:</b> описать кнопку и типы нагрузки, которая она создает.</p> <p>2. Надо скопировать профиль одной из кнопок - той, которая генерит больше всего нагрузки (определить в предыдущем задании). <b>Результат:</b> прислать файл-профиль.</p>	<p>This link is not available, so the page <a href="https://playground.learnqa.ru/chrome/">https://playground.learnqa.ru/chrome/</a> will be tested</p> <p>On this page there is only a button to enter or reload the page. When updating the page, such type of load as scripting prevails, then there is rendering and systems. After clicking on the login button, scripting and idle takes a lot of time.</p> <p>Attached the file for reload page performance <a href="https://drive.google.com/drive/folders/tiIhuPuOuoP4knAvjDzgd9HP3H3YmkYBoB?usp=sharing">https://drive.google.com/drive/folders/tiIhuPuOuoP4knAvjDzgd9HP3H3YmkYBoB?usp=sharing</a></p>		

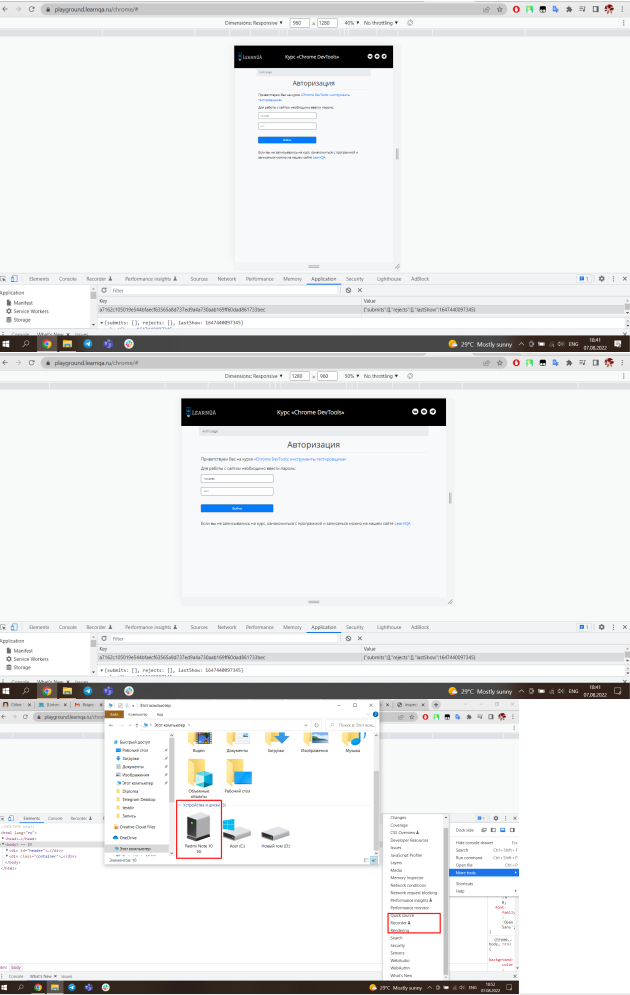
		Открываем <a href="http://arsbatyrov.ru/chrome/application_dz">http://arsbatyrov.ru/chrome/application_dz</a> Мы видим некий сайт, работа которого завязана на данные из Storage и Cookie.		
	Application	<p>1. Cookie могут выставляться как за счет JS, так и за счет сервера. В первом случае JS сам во время работы дает команду на выставление cookie. Во втором обязательно должен быть запрос к серверу, а в запросе в поле Response Headers должен быть заголовок Set-Cookie со значением cookie, который надо выставить. Наша задача - определить, какая кнопка выставляет cookie за счет ответа сервера, а какая - за счет работы самого JS. <b>Результат:</b> написать ответ, какая кнопка каким способом выставляет Cookie.</p>	<p>This link is not available, so the page <a href="https://playground.learnqa.ru/chrome/">https://playground.learnqa.ru/chrome/</a> will be tested</p> 	<p>Since there is no access to the homework page, I am attaching screenshots with cookies stored on the site and in what protocol they were used.</p>
		<p>2. Есть кнопка, которая сохраняет введенные нами текст. Но мы не знаем куда, в cookie, local session или session storage. Наша задача - это понять, используя вкладку Application. <b>Результат:</b> написать ответ на вопрос, куда сохраняется текст (в cookie, local session или session storage).</p>		<p>Session storage is empty and screenshot with cookie has been attached on the previous task. So here is screenshot with local storage.</p>
		<p>3. Описать, как бы вы действовали для решения предыдущей задачи при условии, что Application-вкладки не было бы. Возможно было бы определить, где именно сохраняется информация? Результат: написать ответ.</p>		
		<p>4. Задача со звездочкой. Даже на такой простой вкладке все равно затесалась XSS-уязвимость. Как ее воспроизвести? <b>Результат:</b> описание уязвимости.</p>	<p>You can check the amount of your cookie files (with the help of settings in google chrome) before clicking the button and then after. If number has not changed the text has not been stored in the cookie, and if it has increased, then the text has been stored in the cookie. Also you can check the session storage. Enter your text and then click the button for saving. Then open a new tab, if you can't see your entered text so it was saving only in session storage.</p> <p>Enter js script with alert function in the field and save it with the help of local or session storage. And when you click on save button you will see the alert window. So that demonstrate XSS vulnerability.</p>	
		Открываем <a href="http://arsbatyrov.ru/chrome/device_dz">http://arsbatyrov.ru/chrome/device_dz</a> Видим веб-приложение с меню и полями для ввода. Наша задача - протестировать это приложение.	<p>This link is not available, so the page <a href="https://playground.learnqa.ru/chrome/">https://playground.learnqa.ru/chrome/</a> will be tested</p>	

Device

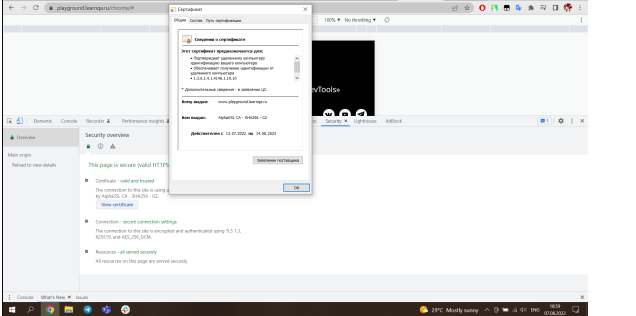
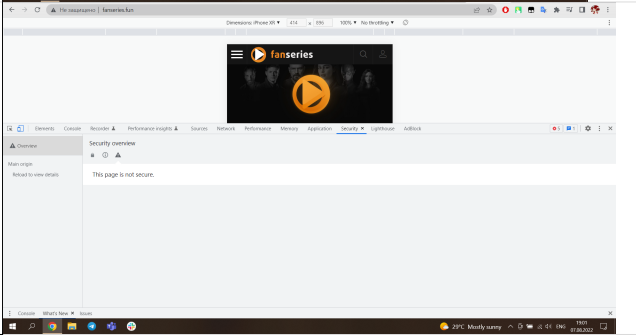
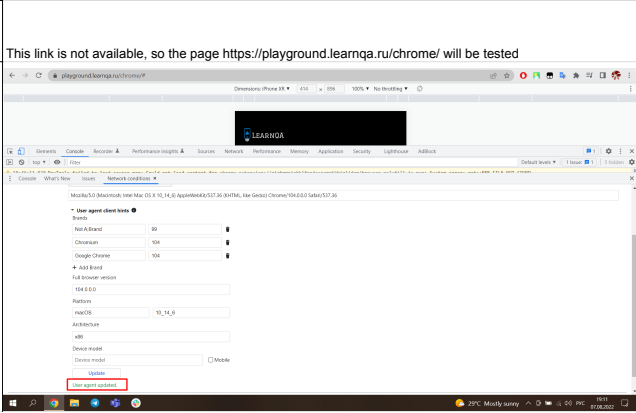
1. Проверить отображение в портретном и ландшафтном режимах на девайсе с разрешением 1280 на 960. **Результат:** два скриншота, по одному для каждого из режимов.

2. Задача со звездочкой. Подключиться к устройству на Android. Открыть веб-приложение из задания в шготе на устройстве. Результат: скриншот экрана компьютера с открытым инспектором.

На этот раз наше задание не связано с сайтом <http://arsbatyrov.ru>

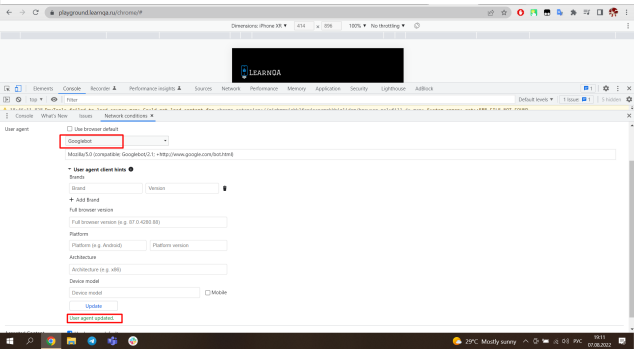


Can't find remote devices option

<div>Security</div>	<div>1. Найти сайт, который защищен SSL-сертификатом. <b>Результат:</b> скриншот деталей его сертификата, который показывает Chrome DevTools.</div>		
	<div>2. Найти сайт, который не защищен SSL-сертификатом. <b>Результат:</b> скриншот предупреждения от Chrome о том, что сайту не стоит доверять свои данные.</div>		
	<div>Открываем <a href="http://arsbalyrov.ru/chrome/networkcond_dz">http://arsbalyrov.ru/chrome/networkcond_dz</a> Мы видим сайт, работа которого зависит от параметра User Agent. Наша задача - протестировать его:</div>	<div>This link is not available, so the page <a href="https://playground.learnqa.ru/chrome/">https://playground.learnqa.ru/chrome/</a> will be tested</div> 	<div>For how long this page is unavailable for work, I am attaching screenshots with the change of user agents.</div>

Network Conditions

2. Надо найти как минимум один User Agent, при котором появляется JS-ошибка. **Результат:** написать значение User Agent.
3. **Задача со звездочкой.** И снова, куда же без полюбившейся нам XSS-уязвимости? Как ее воспроизвести? **Результат:** описание уязвимости.



Maybe XSS vulnerability can appear when hacker puts some js script into custom user agent field.