**Полезные функции DevTools для тестировщиков**

* [Google Chrome](https://habr.com/ru/hub/google_chrome/),
* [Тестирование веб-сервисов](https://habr.com/ru/hub/web_testing/)
* [Из песочницы](https://habr.com/ru/sandbox/)

Всем привет! Меня зовут Миша, я работаю на позиции ручного тестировщика, или Manual QA - кому как удобно. В связи с тем, что в моей работе преобладает ручное тестирование -  я часто сталкиваюсь с консолью разработчика в браузере (думаю как и 99.9% web-тестировщиков).

В интернете огромное количество источников, в которых можно найти информацию про DevTools, как для разработчиков, так и для тестировщиков. Конечно, наполнение таких статей очень сильно разнится в зависимости от ее направленности. Изучив большое количество подобного материала и поняв, что нас (тестировщиков) обделяют информацией :), решил залезть в первоисточник для изучения инструментов разработчика в полном объеме. Пройдясь по всем пунктам огромного меню, выписал для себя порядка 20 пунктов, которые были бы интересны (читай полезны) для тестировщиков. Сразу скажу, что в статье я не буду рассказывать, как пользоваться тем или иным инструментом, так как это подробно описано в статьях, которые будут прикреплены к каждому из пунктов. Цель моего повествования - скорее вычленить из огромного списка возможностей DevTools, именно те, которые были бы полезны для QA-специалистов. Не претендую на объективность и полную раскрытость темы, но постараюсь это сделать.

**P.S.:** Очередность пунктов в списке не говорит об их важности.

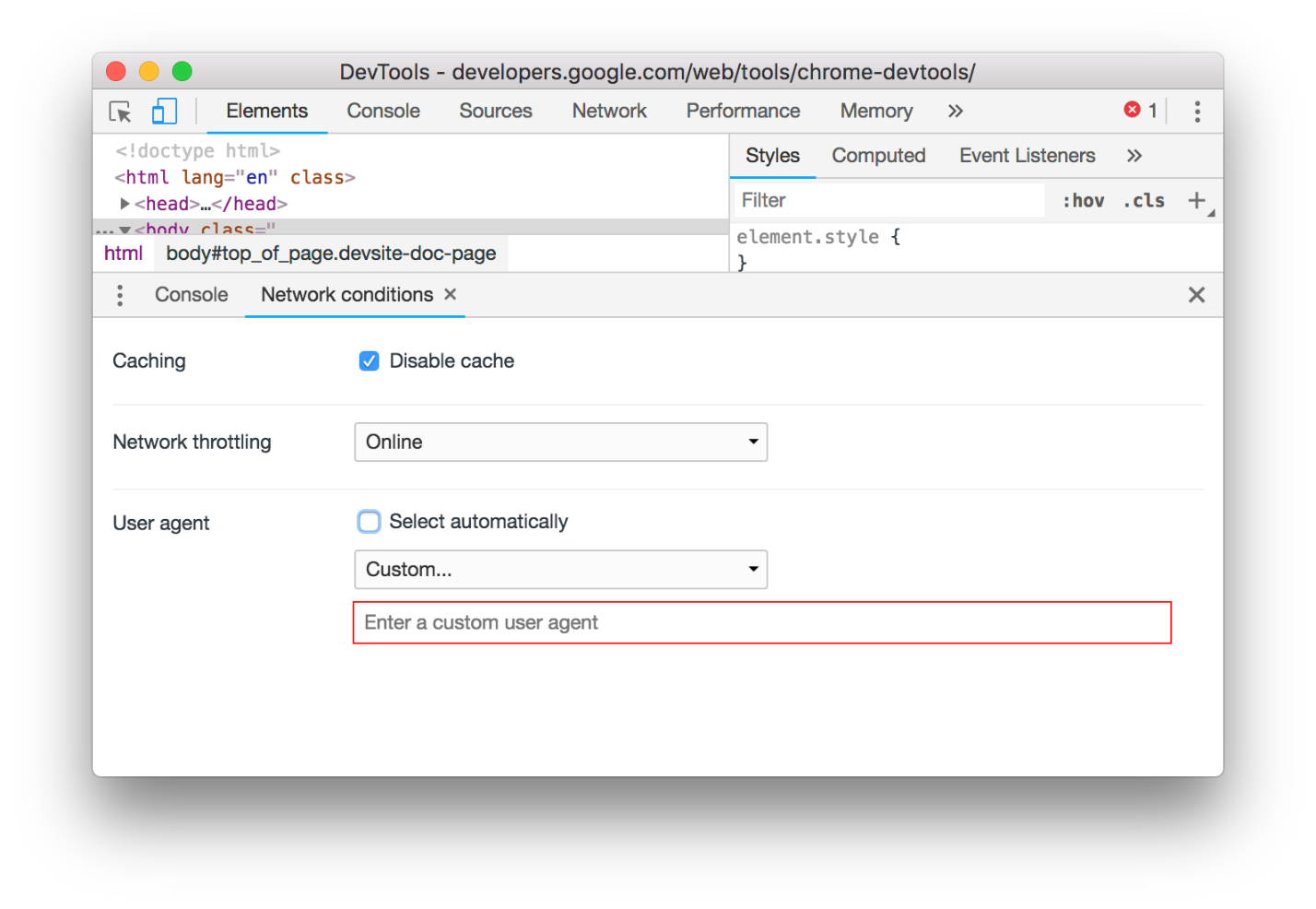
1. [**Эмуляция**](https://developers.google.com/web/tools/chrome-devtools/device-mode/testing-other-browsers) **android и ios, подключение** [**android**](https://developers.google.com/web/tools/chrome-devtools/remote-debugging) **при отладке.**

Как мне кажется, на данный момент большинство компаний, которые делают продукты для мобильных устройств, имеют парк, из необходимых для полного тестирования их продукта, девайсов. Однако далеко не все считают необходимым тратить деньги на подобное. И, необходимость протестировать свой продукт на том или ином устройстве, не всегда зависит от того делают ли данный продукт для мобилок или нет. В связи с этим есть необходимость проверять свои сайты на мобильных устройствах без их физического присутствия. Минус данного подхода заключается в том, что большинство этих эмуляторов являются коммерческими.

Второй подпункт позволяет без всяких эмуляторов отследить все запросы и поведение Вашего продукта на устройствах android, просто подключив его к компьютеру и произведя небольшое количество манипуляций. Также плюс этого способа заключается в том, что можно настроить доступ к локальным серверам через такой тип подключения.

1. **Переопределение** [**геолокации**](https://developers.google.com/web/tools/chrome-devtools/device-mode/geolocation) **и подмена** [**User-Agent**](https://developers.google.com/web/tools/chrome-devtools/device-mode/override-user-agent).

Продолжим рассматривать возможности DevTools для мобильных устройств. В вышеуказанных двух пунктах говорится о возможности изменять (подменять) геолокацию нахождения устройства и параметры юзер агента. Думаю, что многим тестировщикам частенько приходится воспроизводить какие-либо баги, которые были выловлены клиентами продукта не имея на то соответствующих технических возможностей. Подмена User-Agent поможет воспроизвести тот или иной баг, который был воспроизведен из какой-либо версии браузера или ОС. Закончив тестирование, никогда не забывайте возвращать данные User-Agent в исходное положение.

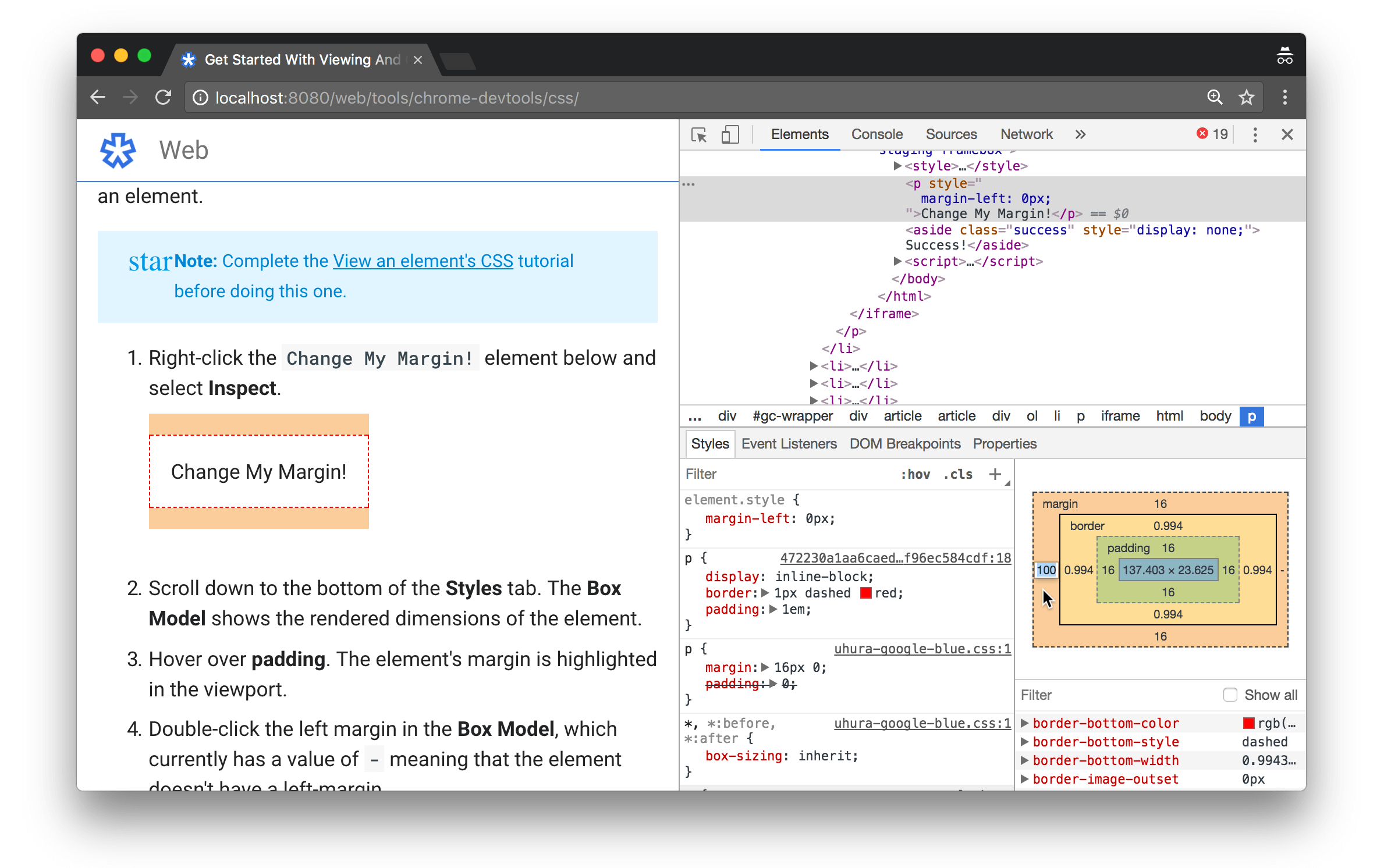
Подмена User-Agent

1. **Определение** [**JS пути**](https://developers.google.com/web/tools/chrome-devtools/dom#path) **к строке.**

Этот пункт будет больше интересен тем, кто занимается автоматизацией тестирования. Скопировав полный путь к определенной строке в формате JS, можно ссылаться на него в автоматизированном тесте. Безусловно, данный способ не самый популярный для автоматизаторов, потому что этот путь может часто меняться, но на первых порах, когда еще не будет выработан скилл, помогающий с закрытыми глазами строить нужные селекторы для тестов, то эта возможность в DevTools может Вам помочь.

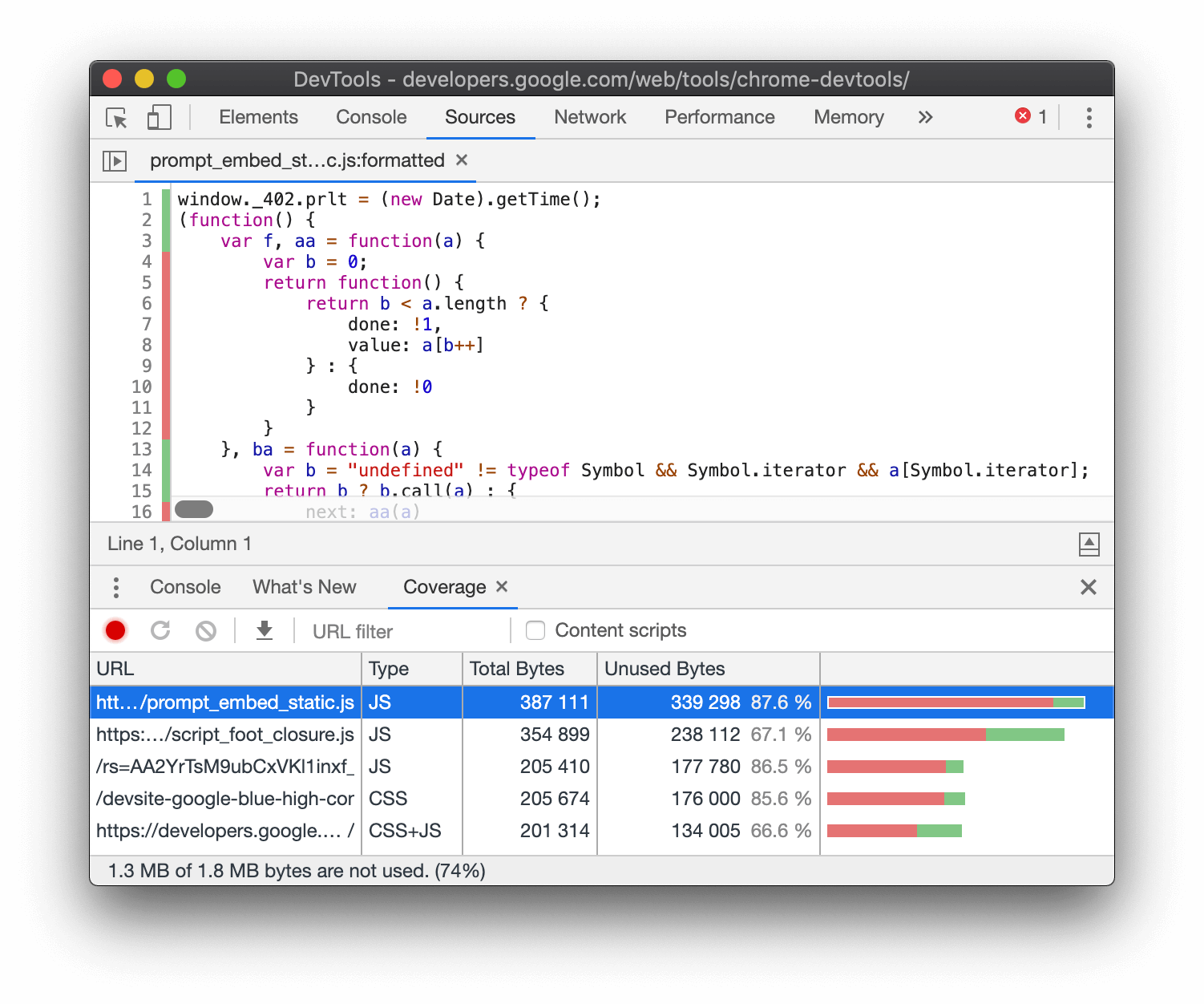
1. [**Изменение стилей CSS**](https://developers.google.com/web/tools/chrome-devtools/css#declarations) **у элементов.**

Считаю очень полезным умением для тестировщика представлять, как может выглядеть та или иная кнопка на сайте или какое-либо поле. В данном пункте рассматривается добавление фонового окраса для поля. Помимо этого, для элементов можно изменять и другие параметры (шрифт, размер, цвет и т.д.), для того чтобы можно было сразу указать разработчику или дизайнеру, как Вы видите этот элемент в контексте страницы либо, по просьбе заказчика изменить кнопку в “live” режиме.

Пример изменения размера поля элемента

1. [**Неиспользуемые CSS и Javascript**](https://developers.google.com/web/tools/chrome-devtools/coverage) **в вёрстке.**

Не будем забывать про тестирование производительности, данный пункт будет интересен именно с точки зрения ускорения загрузки Вашего web-сайта. Если количество неиспользуемого кода, который каждый раз “пробегает” при загрузке той или иной страницы, очень велико, то при помощи действий, описанных в статье, будет возможность найти весь неиспользуемый код и указать его, как артефакт в баг репорте.

Отчет о покрытии кода

1. **Немного интересного про** [**debug JavaScript**](https://developers.google.com/web/tools/chrome-devtools/javascript)**.**

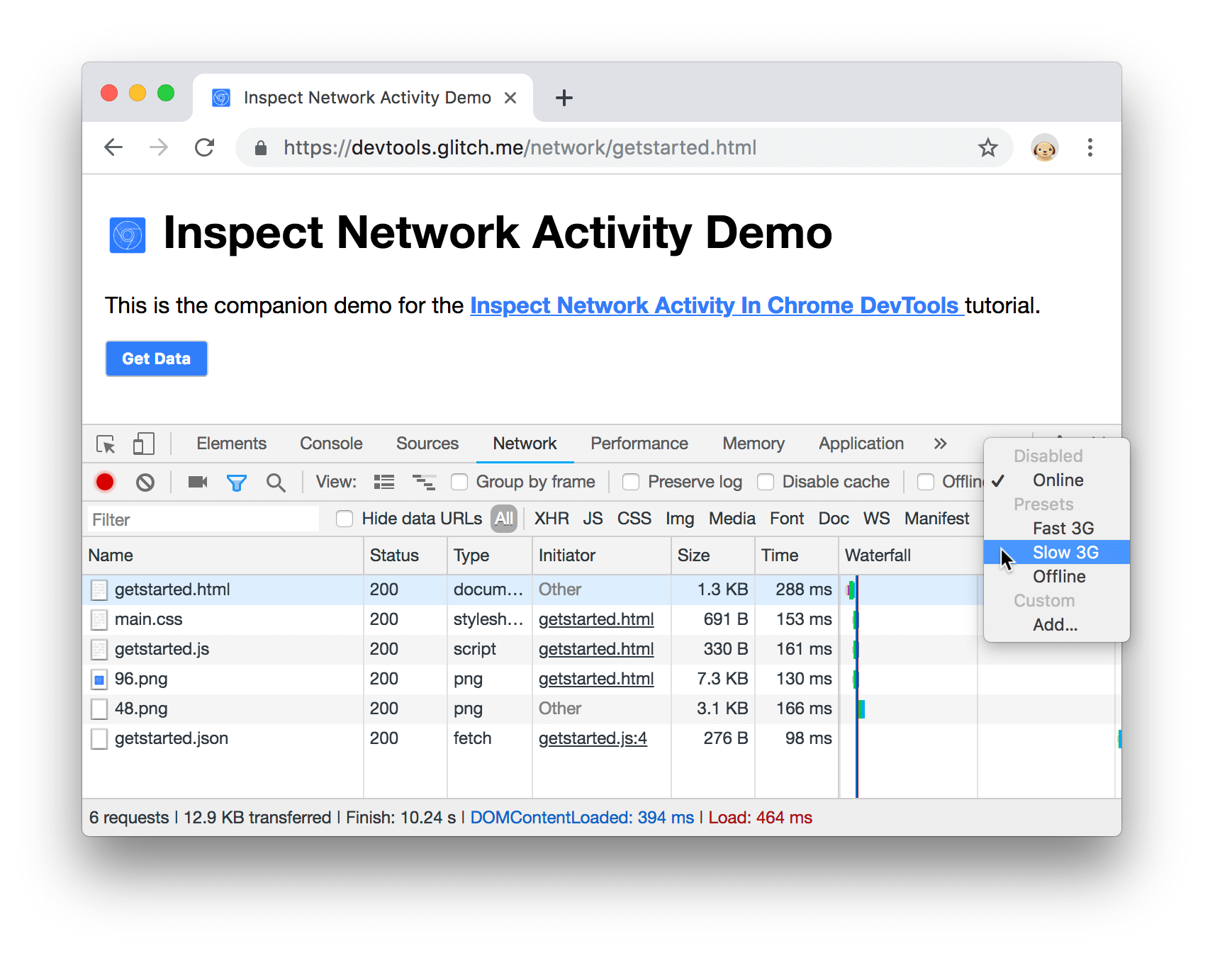
Многим этот пункт покажется лишним, ведь дебажить код - это вещь, которой в основном занимаются разработчики для отладки кода. Но со своей стороны хочу сказать, что это умение для тестировщика, лишним точно не будет. Безусловно для отладки кода необходимо уметь его читать. Думаю, многие видели большое количество мемов про JavaScript, которые красноречиво говорят о том, что язык далеко не самый легкий, особенно для простого обывателя. С другой стороны, некоторые данные иногда формируются на фронте, проходя несколько функций и понять почему в переменной сформировалось определенное значение, бывает очень важно. Даже в мои 3 года опыта работы в тестировании я уже сталкивался с дебагом кода именно в Chrome. Безусловно, без помощи разработчика я бы вряд ли смог это сделать, но в этой ситуации я понял, что этот момент очень важен.

1. [**Сохранение изменений в Chrome**](https://developers.google.com/web/updates/2018/01/devtools#overrides) **при перезагрузке страницы**.

Такую возможность добавили в DevTools относительно недавно (с 65 версией). Она позволяет сохранять все изменения, которые были внесены в те же CSS стили, о которых я говорил выше. И при перезагрузке страницы они сохранятся, чтобы, например, была возможность посмотреть, как ведет себя измененная кнопка при загрузке страницы.

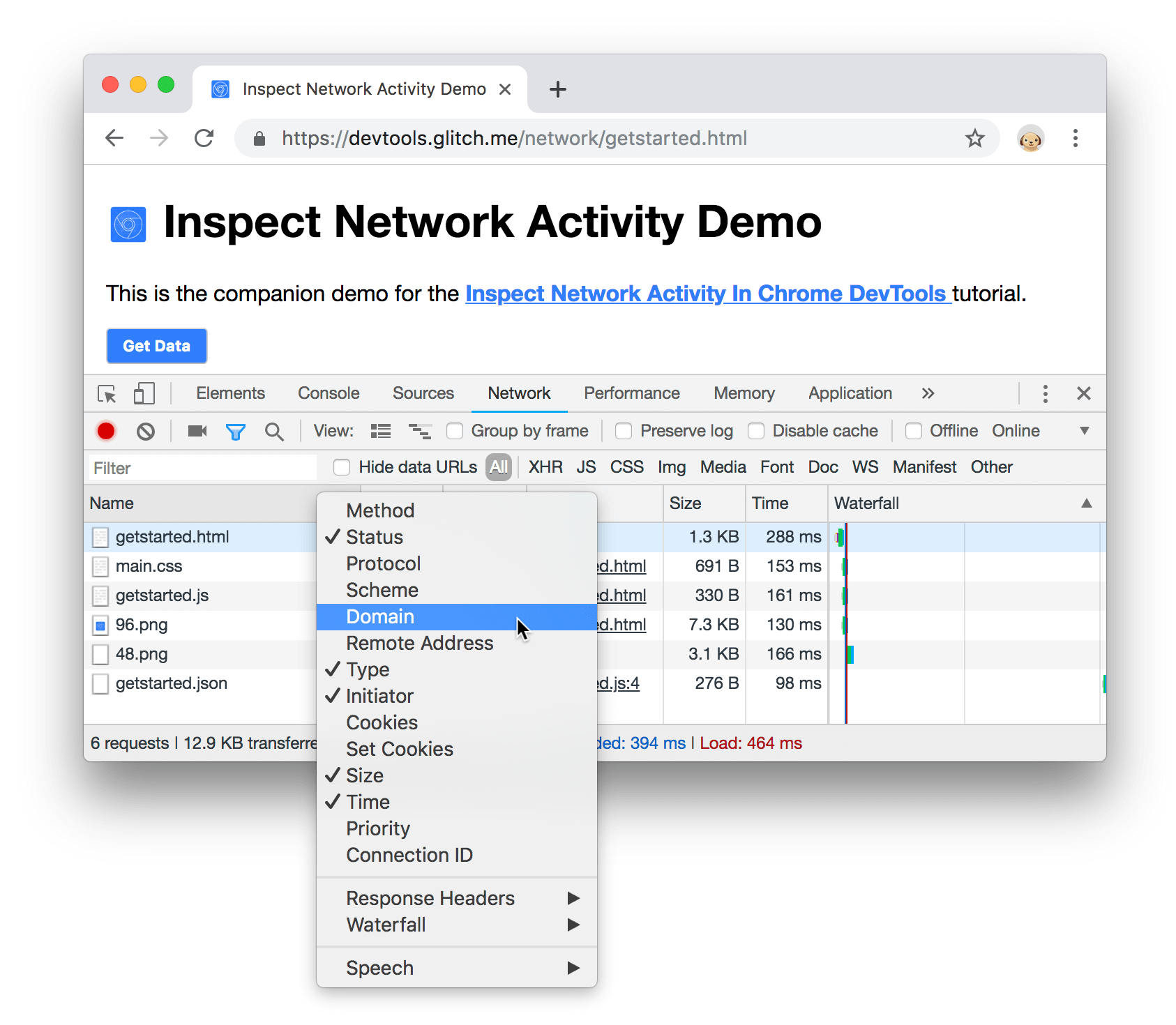
1. [**Имитация медленного сетевого соединения**](https://developers.google.com/web/tools/chrome-devtools/network#throttle).

Эффект DevTools, который демонстрирует, как ведет себя страница при её загрузке на мобильных устройствах. Разработка, в 90% случаев, ведется при хороших условиях, связанных со скоростью интернета и его стабильностью, также как и тестирование, в связи с чем выловить баг, который воспроизводится у пользователя, становится возможным только полностью воспроизведя все окружение, в котором этот баг проявился. С появлением нового высокоскоростного интернета для мобильный устройств, возможно эта проблема будет не совсем актуальной, но пока этого не происходит. По крайней мере не во всех странах  .

Выбор скорости соединения

1. **Настройка столбцов** [**на вкладке Network**](https://developers.google.com/web/tools/chrome-devtools/network#information)**.**

Тоже очень полезная вещь, которую я не смог найти ни в одной статье про DevTools для тестировщиков. Здесь можно настроить именно те столбцы, которые необходимы для анализа запросов на сайте в Вашем конкретном случае.

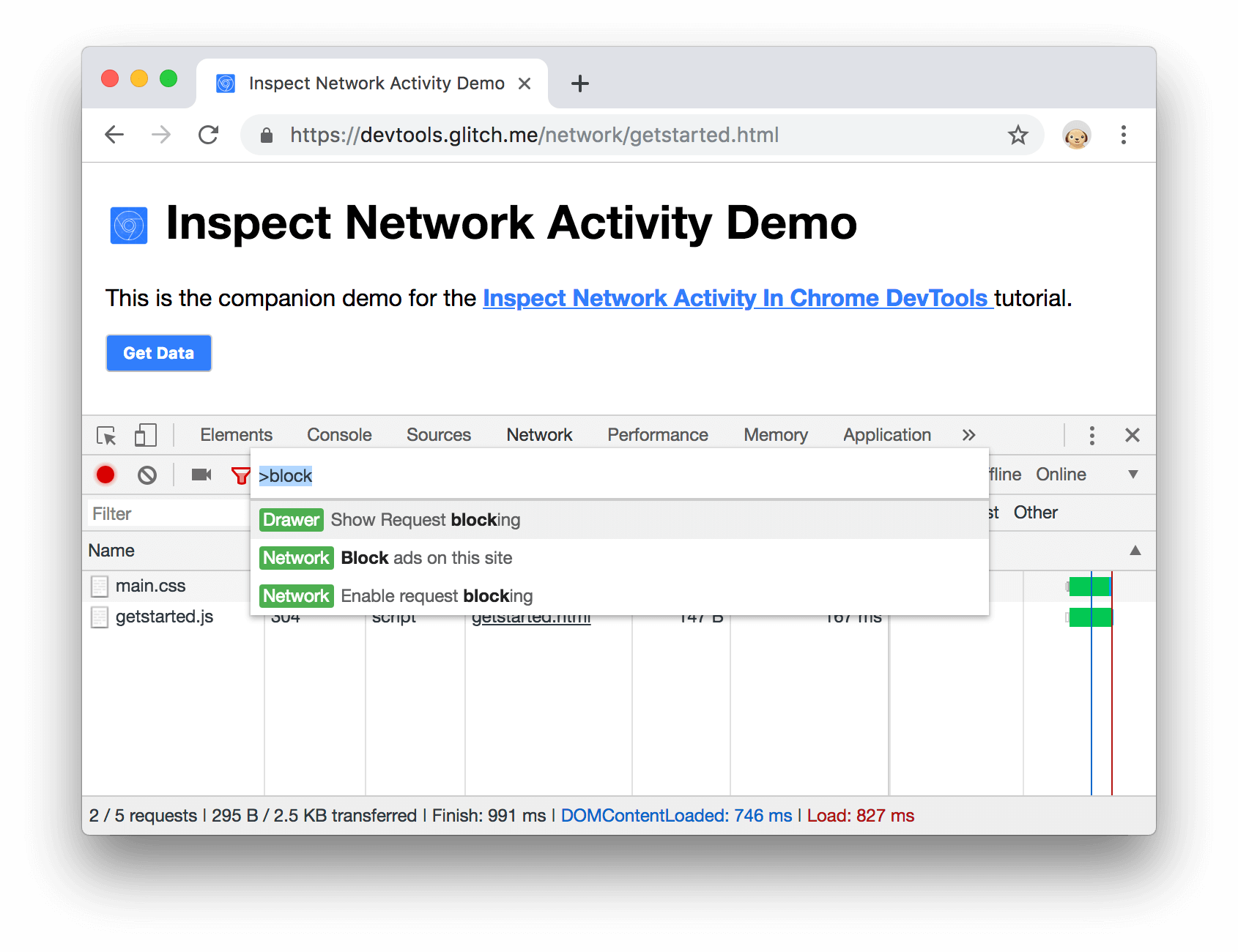
Список возможных столбцов в Network

1. [**Снимки экрана**](https://developers.google.com/web/tools/chrome-devtools/network/reference#screenshots) **при загрузке страницы.**

Думаю, этот пункт было бы логично связать с восьмым в этом списке. Тоже очень полезная вещь, как мне кажется, которая может помочь уловить плавающий баг, либо отследить ненормальное или нелогичное поведение во время загрузки страницы.

1. [**Блокирование запросов**](https://developers.google.com/web/tools/chrome-devtools/network#block)**.**

С развитием рекламы в сети появились и различные приложения, которые эту рекламу блокируют, для удобства пользования браузерами. Для того, чтобы проверить как себя поведет страница, если будет заблокирован тот или иной запрос, можно воспользоваться блокировкой интересующего запроса, хороший пример описан в источнике.



1. [**Все о cookies**](https://developers.google.com/web/tools/chrome-devtools/storage/cookies) **во вкладке applications**.

Cookies - очень важная вещь для анализа пользовательских сессий, соответственно, так как мы (тестировщики) воспроизводим пользовательские сценарии, нам необходимо знать, как можно улучшить тест-кейсы, используя работу с куками. В статье описаны все сценарии (поиск, удаление, изменение данных).

**Бонусы:**

Здесь я бы хотел оставить те ссылки (с небольшими пометками), которыми я лично еще не пользовался, но которые, по моему мнению, были бы полезны для изучения и последующего применении тестировщиком на практике:

* [описание всех shortcuts, которые существуют в chrome DevTools](https://developers.google.com/web/tools/chrome-devtools/shortcuts);
* [все об анализе безопасности](https://developers.google.com/web/tools/chrome-devtools/security);
* [имитация нового посетителя](https://developers.google.com/web/tools/chrome-devtools/network/reference#disable-cache);
* [повтор xhr запроса](https://developers.google.com/web/tools/chrome-devtools/network/reference#replay-xhr);
* [проверка анимации](https://developers.google.com/web/tools/chrome-devtools/inspect-styles/animations);
* [основные команды для запуска JavaScript в консоли](https://developers.google.com/web/tools/chrome-devtools/console/javascript);
* [основные проблемы, которые можно увидеть во вкладке network](https://developers.google.com/web/tools/chrome-devtools/network/issues).

Безусловно DevTools не ограничивается тем функционалом, который я описал выше. Есть очень интересные вкладки, которые называется performance и audit, но я не стал нагружать еще этой информацией, так как считаю это темой для отдельной статьи, если в целом это интересно будет прочитать и познакомиться с этими вкладками в DevTools.