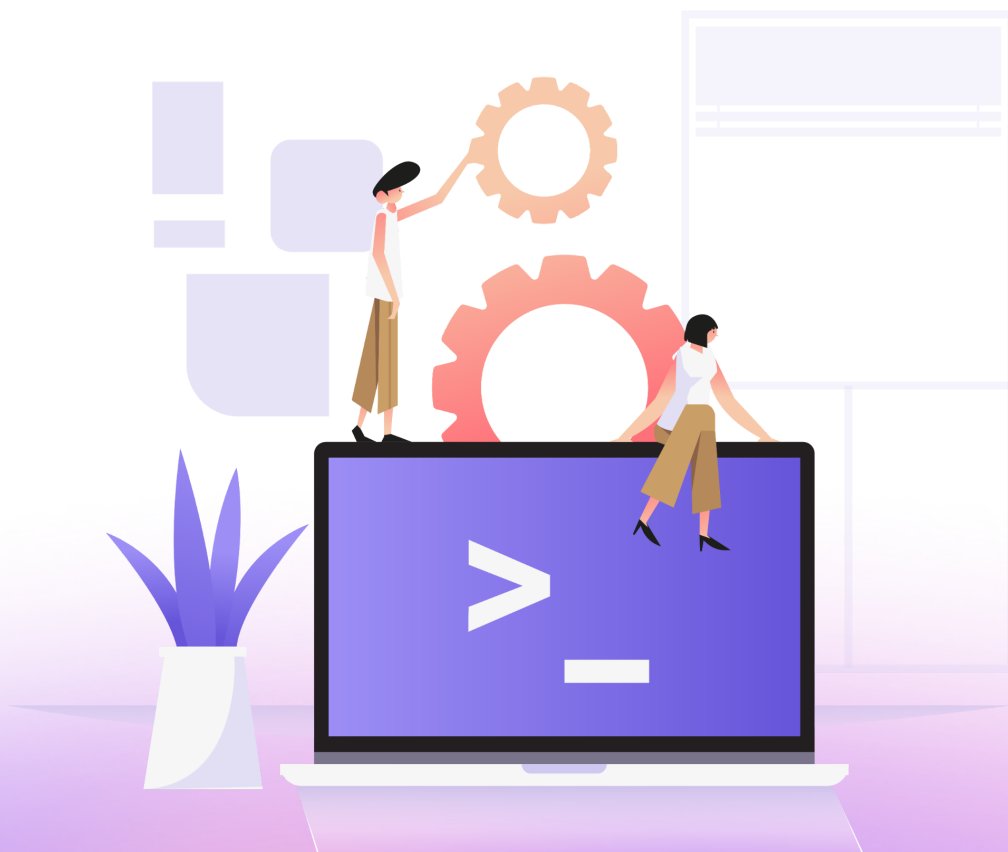


Тестирование веб-приложений

# Инструменты разработчика в Chrome



# На этом уроке

1. Узнаем, что такое инструменты разработчика.
2. Научимся анализировать веб-приложение.
3. Выясним, как обнаруживать некоторые типы ошибок, используя инструменты разработчика.

## Оглавление

[Обзор Google Chrome DevTools](#)

[Структура DevTools](#)

[Device Toolbar](#)

[Elements](#)

[Выбор элемента для работы](#)

[Работа с конкретным элементом](#)

[Активация псевдоклассов](#)

[Наглядное изменение стилей](#)

[Console](#)

[Sources](#)

[Network](#)

[Время загрузки страницы](#)

[Режим диафильма](#)

[Фильтры](#)

[Профили сети](#)

[Performance](#)

[Memory](#)

[Application](#)

[Security](#)

[Lighthouse](#)

[Практическое задание](#)

[Глоссарий](#)

[Дополнительные материалы](#)

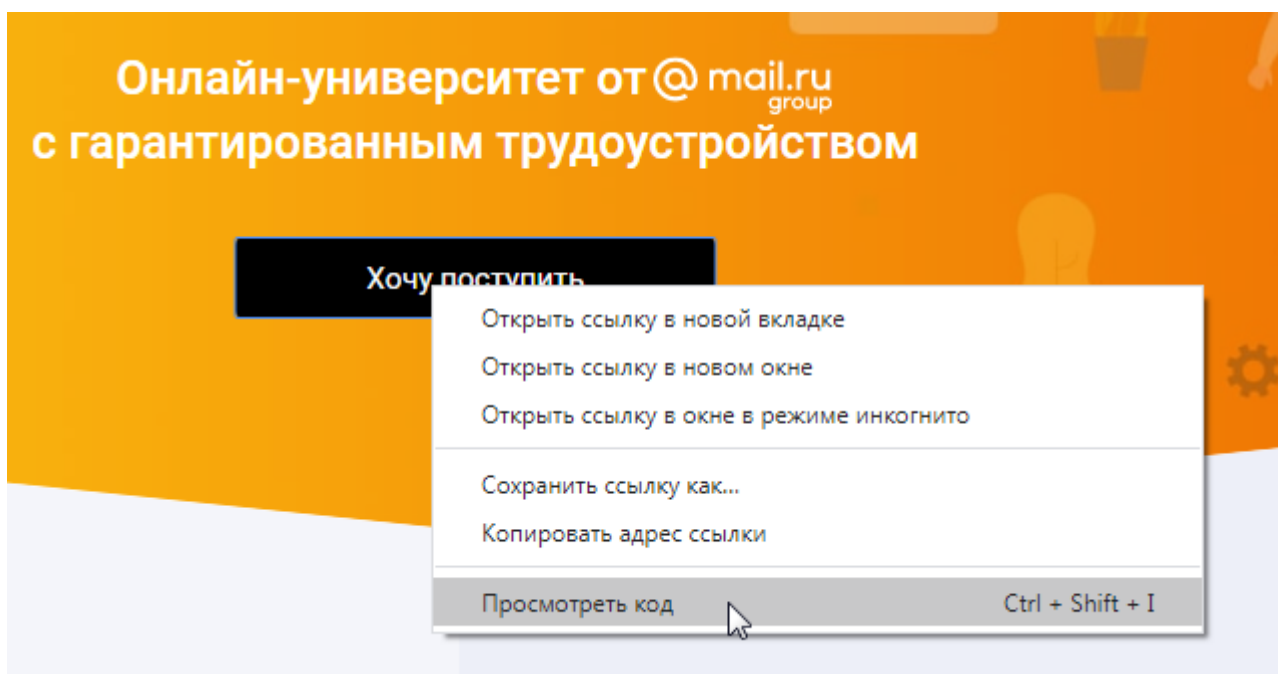
[Используемые источники](#)

# Обзор Google Chrome DevTools

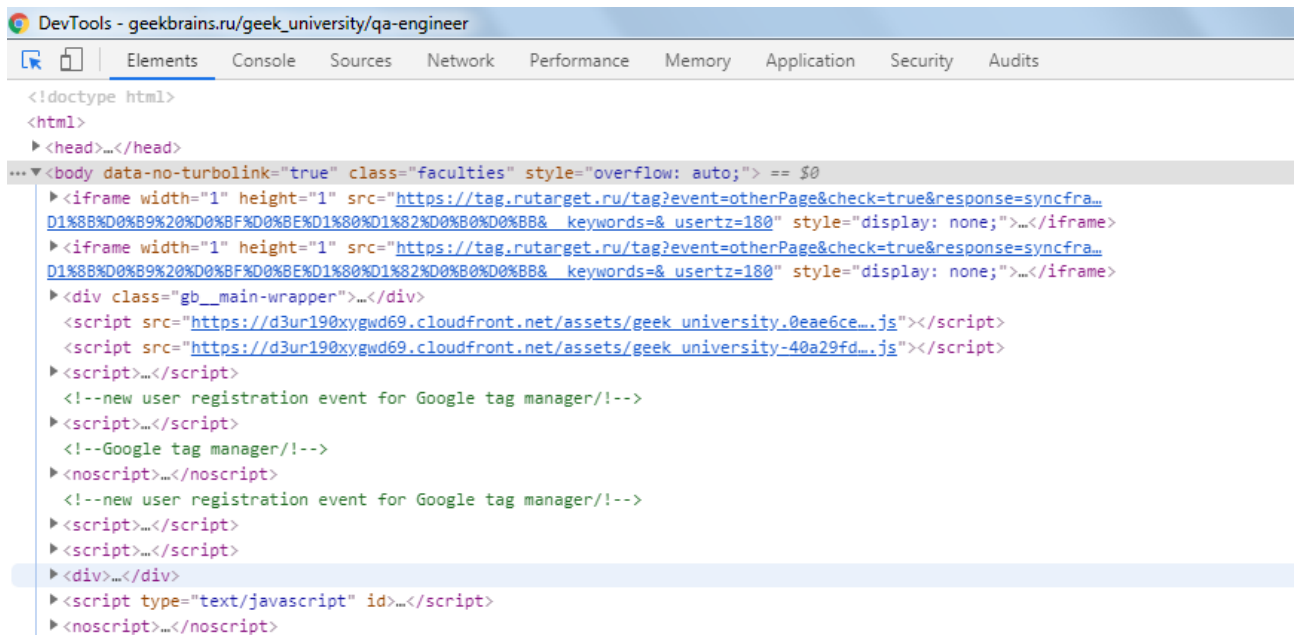
Google Chrome — браузер, разрабатываемый компанией Google на основе браузера с открытым исходным кодом Chromium. Как мы видели на предыдущей лекции, Google Chrome — это самый востребованный браузер. Среди веб-разработчиков и тестировщиков он также наиболее популярен, потому что Google Chrome снабжён хорошим набором инструментов для разработки и тестирования веб-приложений — **Chrome DevTools**. Этот набор инструментов позволяет тестировать, отлаживать, профилировать, проверять код на соответствие стандартам и делать многое другое.

Есть несколько способов открыть Chrome DevTools в зависимости от того, к какому инструменту мы хотим получить доступ.

1. Если надо проверить стили или атрибуты узла DOM, щёлкаем элемент правой кнопкой мыши и выбираем «Просмотреть код».

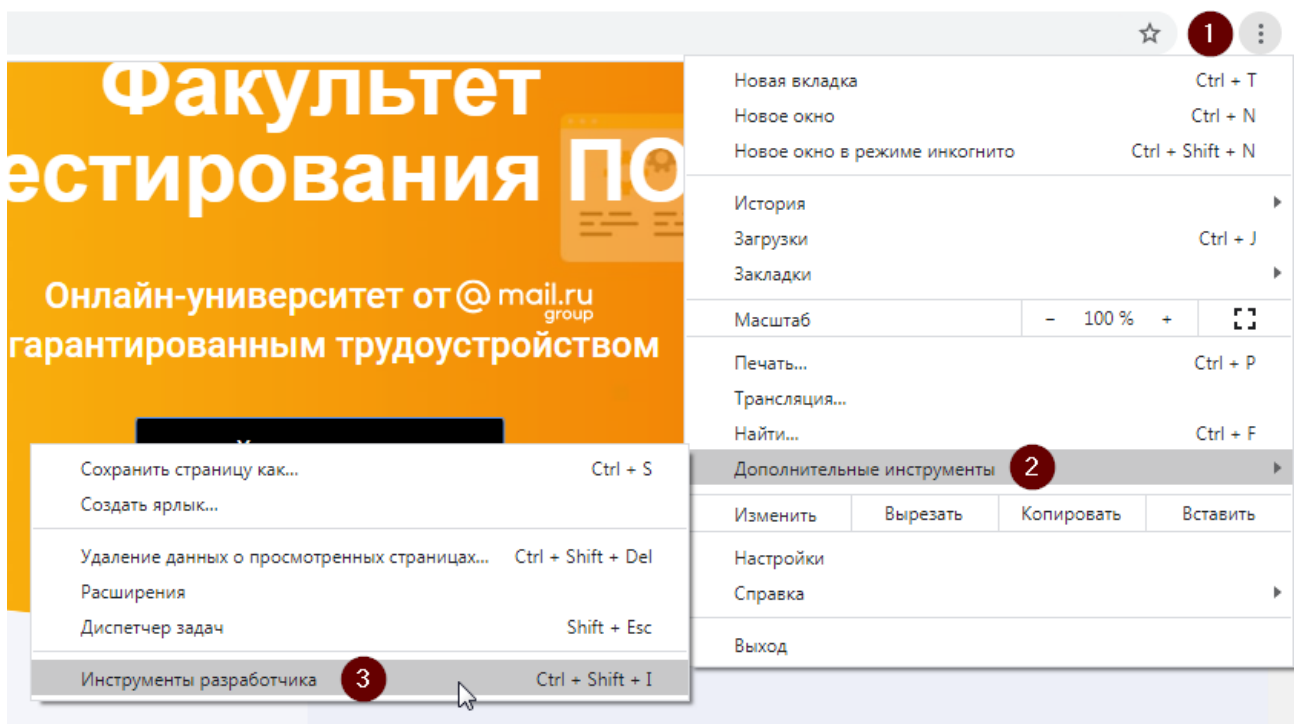


## 2. Комбинация клавиш Ctrl+Shift+I или F12:



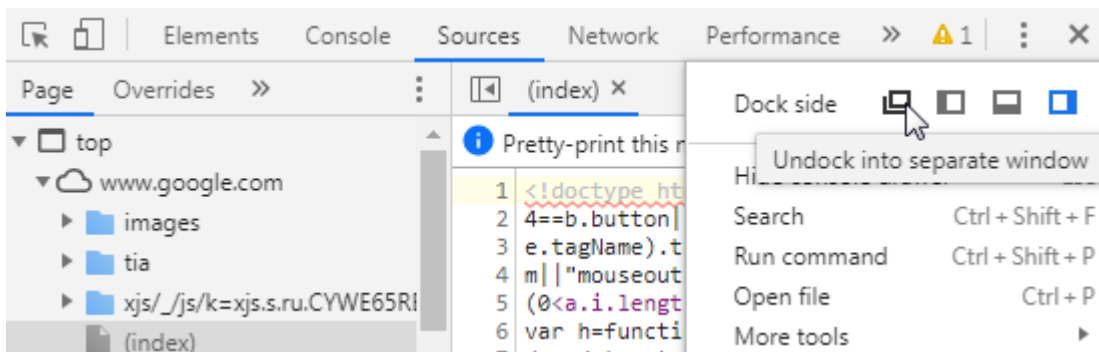
```
<!doctype html>
<html>
  <head>...</head>
  <body data-no-turbolink="true" class="faculties" style="overflow: auto;"> == $0
    <iframe width="1" height="1" src="https://tag.rutarget.ru/tag?event=otherPage&check=true&response=synfra...
    <iframe width="1" height="1" src="https://tag.rutarget.ru/tag?event=otherPage&check=true&response=synfra...
    <div class="gb__main-wrapper">...</div>
      <script src="https://d3ur190xygwd69.cloudfront.net/assets/geek_university.0eae6ce...js"></script>
      <script src="https://d3ur190xygwd69.cloudfront.net/assets/geek_university-40a29fd...js"></script>
      <script>...</script>
      <!--new user registration event for Google tag manager!-->
      <script>...</script>
      <!--Google tag manager!-->
      <noscript>...</noscript>
      <!--new user registration event for Google tag manager!-->
      <script>...</script>
      <script>...</script>
    <div>...</div>
    <script type="text/javascript" id>...</script>
    <noscript>...</noscript>
```

## 3. Через основное меню Chrome:



При переходе в инструменты разработчика появится панель для работы с веб-страницей. По умолчанию она открывается в том же окне справа от основного содержимого, но панель может располагаться и слева, и внизу. Можно расположить панель в отдельном окне и, например, перенести на второй монитор.

Варианты расположения, слева направо:

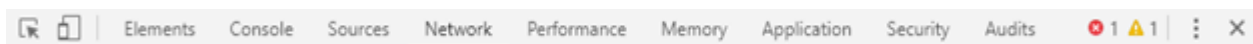


1. Открепить DevTools и открыть их в другом окне.
2. Прикрепить к левому краю.
3. Прикрепить к нижнему краю.
4. Прикрепить к правому краю.

Панель открывается только для текущей вкладки.

## Структура DevTools

В DevTools вкладки самого верхнего уровня называются «Панели»:



Слева, перед панелями, располагаются кнопки выбора элемента на странице и выбора устройства для отображения:



Панель **Elements** используется для выбора и редактирования любых HTML-элементов на странице. Позволяет свободно манипулировать DOM и CSS.

Панель **Console** применяется для логирования диагностической информации в процессе разработки или взаимодействия с JavaScript на странице.

Панель **Sources** показывает все файлы, подключённые к текущей странице.

Панель **Network** позволяет мониторить процесс загрузки страницы и всех файлов, которые подгружаются при её загрузке.

Панель **Performance** отображает таймлайн использования сети, выполнения JavaScript-кода и загрузки памяти, применяется для оценки производительности работы страницы в целом.

Панель **Memory** используется для отслеживания нагрузки, которую оказывает выполнение JS-кода на систему.

Панель **Application** содержит инструменты для просмотра всех загруженных ресурсов и работы с ними: cookie, кеша приложения, изображений, шрифтов, таблиц стилей, локального хранилища.

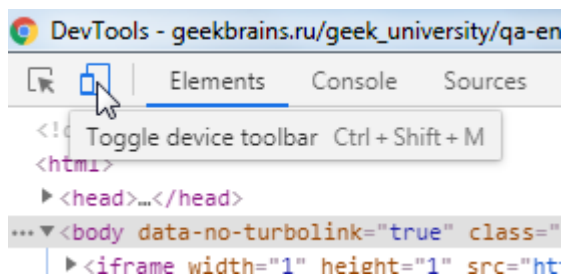
На панели **Security** отображается информация о протоколе безопасности, данные о сертификате, если он есть, безопасной загрузке ресурсов. Инструмент используется для обнаружения проблем безопасного или небезопасного содержимого, проблем сертификатов и так далее.

Панель **Lighthouse**: после выбора параметров и запуска системы аудита происходит анализ загружаемой страницы и предоставляются предложения по оптимизации страницы для уменьшения времени её загрузки и увеличения отзывчивости.

## Device Toolbar

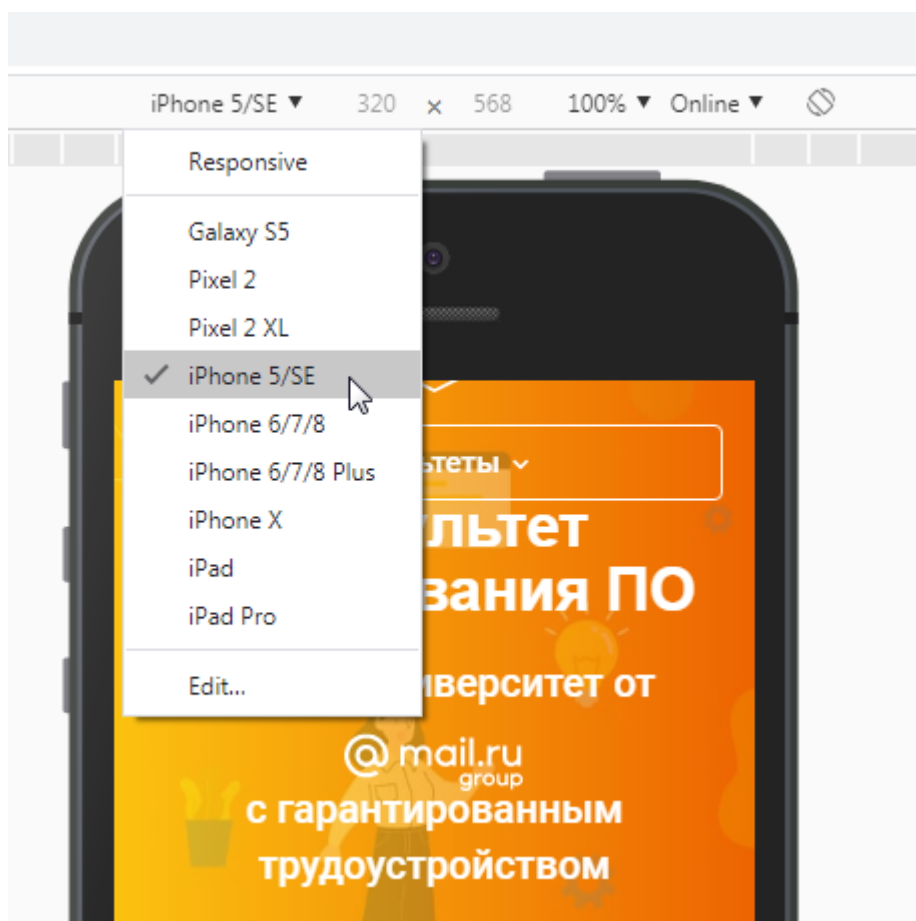
Функция Device Toolbar служит для эмуляции внутри браузера различных разрешений экрана и разных устройств, например, телефонов или планшетов. Они используются для базового тестирования совместимости сайта с разными устройствами.

Активация Device Toolbar происходит при клике на кнопку Device Toolbar или комбинацией клавиш Ctrl+Shift+M:

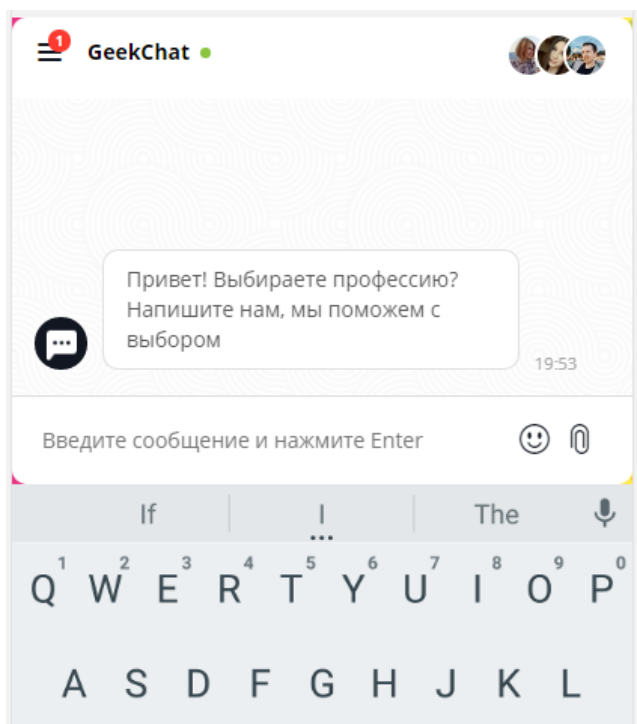


В Chrome есть готовые пресеты для разных мобильных устройств, но также можно добавить собственное, указав размер экрана.

Режим выбора устройства Device Toolbar:

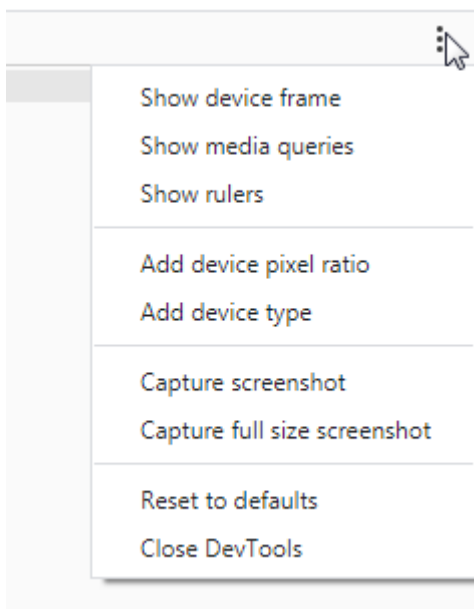


Режим просмотра на разных устройствах позволяет быстро проверить, как страница будет выглядеть на мониторах с разным разрешением или на мобильных телефонах. Функция выбора устройства пригодится при разработке адаптивных веб-интерфейсов, мобильных версий сайтов или для тестирования страниц на разных разрешениях монитора. Для мобильных устройств также эмулируется сенсорный интерфейс, доступна экранная клавиатура для устройства Nexus 5X, ведь вид экрана меняется при включённой клавиатуре. Клавиатура, правда, неактивная — просто изображение.



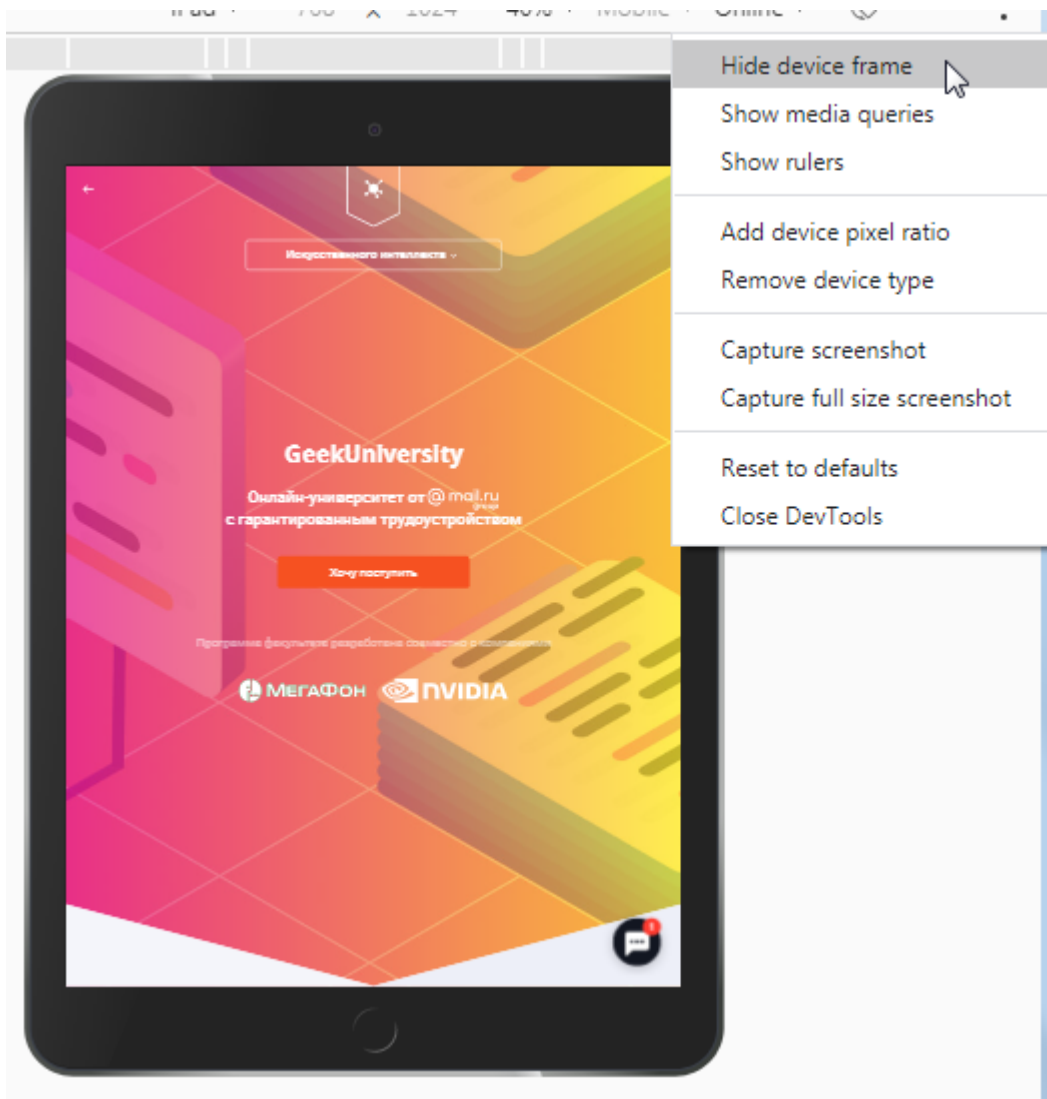
Разные режимы просмотра полезны при тестировании приложений, имеющих разный инструментарий для веб-версии и настольного компьютера. Например, Instagram при просмотре с компьютера не даёт публиковать сообщения, но при переключении в DevTools на режим мобильного телефона такая возможность появляется.

У режима есть собственные настройки. Кнопка с меню располагается прямо под кнопкой основного меню:



Show/Hide device frame включает и отключает отображение рамки для некоторых устройств. Например, iPad:





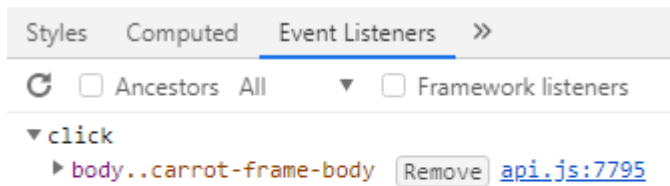
Далее переходим к следующей панели, Elements.

## Elements

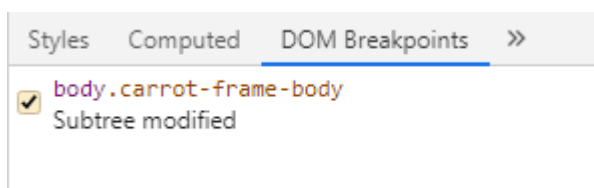
При активном пункте меню Elements можно видеть всё DOM-дерево веб-страницы, изменять свойства HTML-элемента и отслеживать изменения на веб-странице без перезагрузки. При выборе любого DOM-элемента на вкладке Styles отобразятся все CSS-правила, применяемые к нему, в том числе и неактивные. Все правила разбиты на блоки и упорядочены по убыванию специфичности селектора. Можно на лету менять значения, деактивировать и дописывать новые правила и смотреть, как это влияет на отображение. Для выбранного элемента DOM доступно ещё несколько вкладок:

1. **Event Listeners** — содержит все события, относящиеся к конкретному элементу. Например, у активного элемента прописывается выполнение определённого JavaScript-кода при нажатии

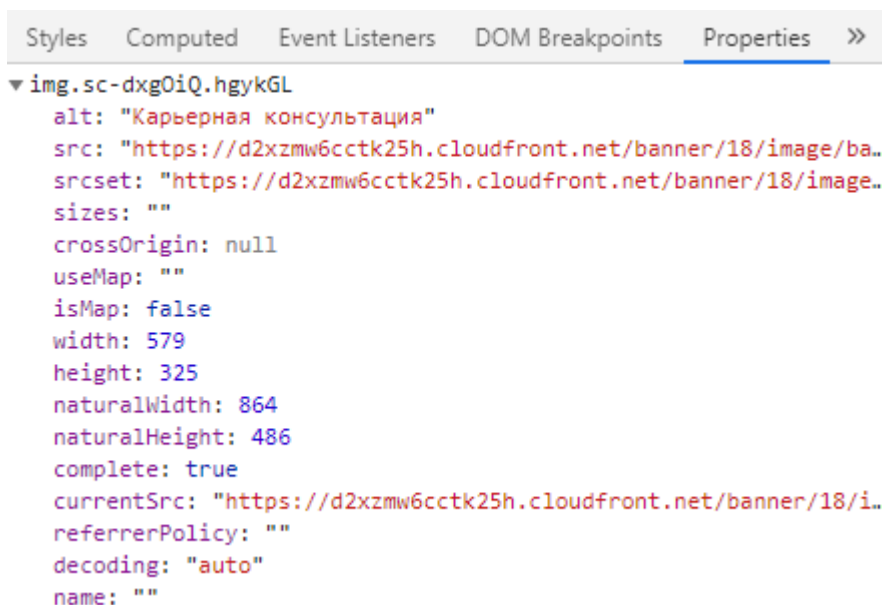
на элемент. Тогда такое событие будет описано как click в этой вкладке:



2. **DOM Breakpoints** — точки останова для элемента. Используются программистами для отладки JavaScript-кода. Программист может добавить остановку выполнения скрипта при изменении DOM, затрагивающем этот элемент: изменении атрибутов этого элемента, дочерней структуры DOM или при полном удалении элемента:



3. **Properties** — список всех свойств элемента:



Возникают также дополнительные вкладки, добавляемые расширениями для Chrome.

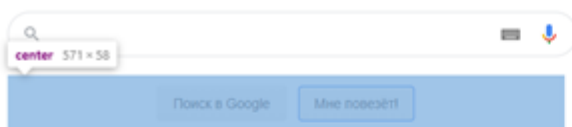
### Ключевые возможности

1. Просмотр и редактирование в live-режиме любого элемента DOM.
2. Просмотр и изменение CSS-правил, применяемых к любому выбранному элементу в панели Styles.
3. Просмотр всего списка событий и свойств для элемента на соответствующих вкладках.

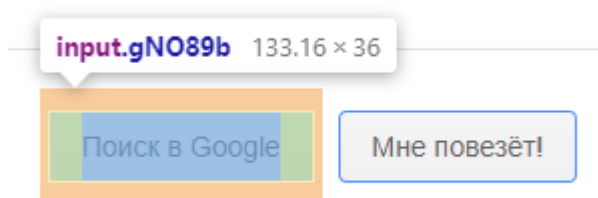
# Выбор элемента для работы

Способы поиска подходящего элемента:

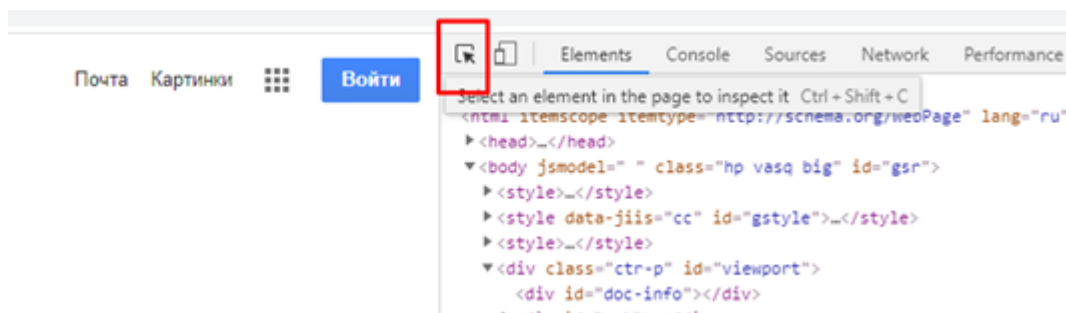
1. При наведении курсора на код соответствующий элемент на самой странице автоматически подсвечивается, а также указывается название тега и его размер в пикселях:



Если у элемента есть внутренние и внешние поля, они выделяются цветом:

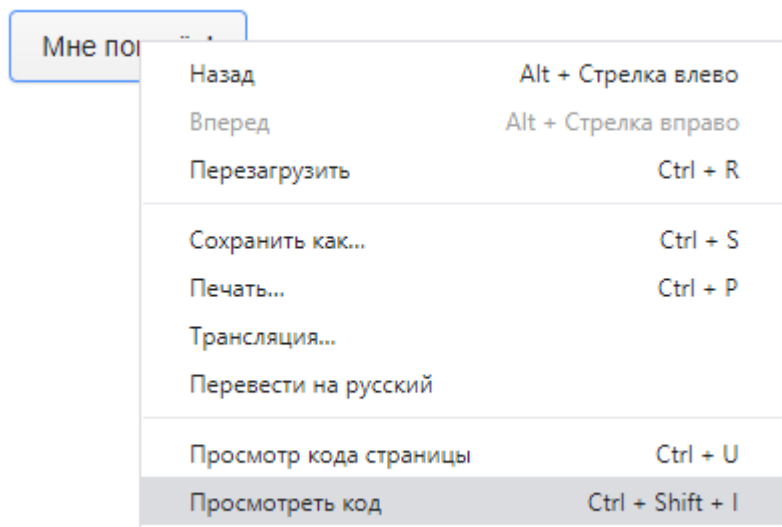


2. Можно выбрать режим «Поиск элемента на странице», нажав Ctrl+Shift+C или на иконку:



Когда этот режим активен, при перемещении курсора по веб-странице в панели разработчиков автоматически показывается искомый элемент.

3. Через контекстное меню: на любом компоненте веб-страницы кликаем правой кнопкой мышки и выбираем «Просмотреть код»:



4. Поиском по DOM-дереву. На панели инструментов нажимаем Ctrl+F и вводим строку для поиска, например, тег, CSS-класс или XPath:

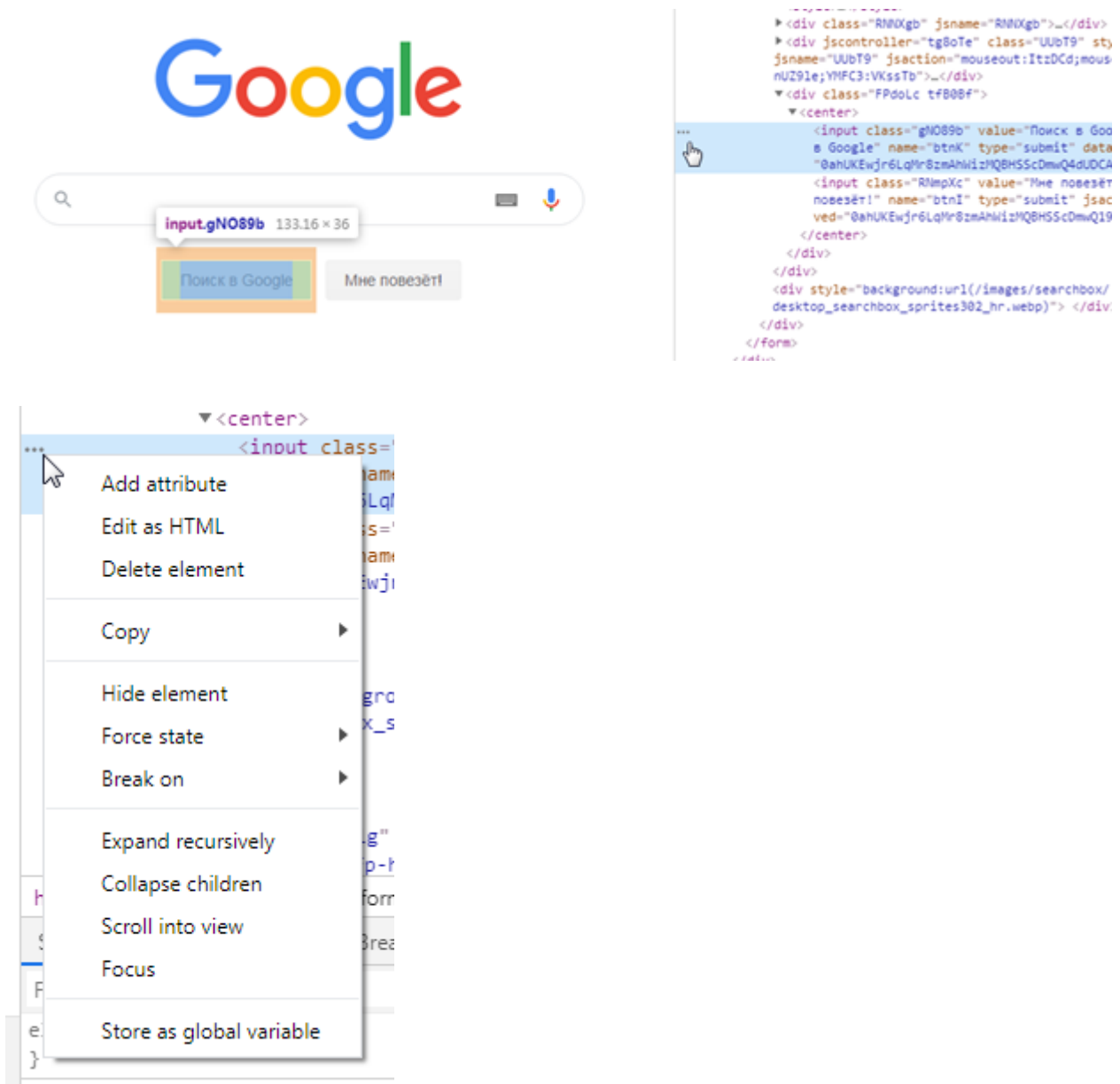


## Работа с конкретным элементом

Выбрав конкретный элемент, можно менять его свойства и смотреть, как это влияет на страницу.

Например, так меняется имя тега. Закрывающий тег автоматически изменится. Аналогично меняются названия классов, параметры высоты и ширины, отступы — все поля доступны для редактирования. Изменения применяются сразу после клика вне тега.

Дополнительные функции редактирования доступны при клике на три точки слева от выбранного элемента. Аналогичные — по правому клику на элемент:

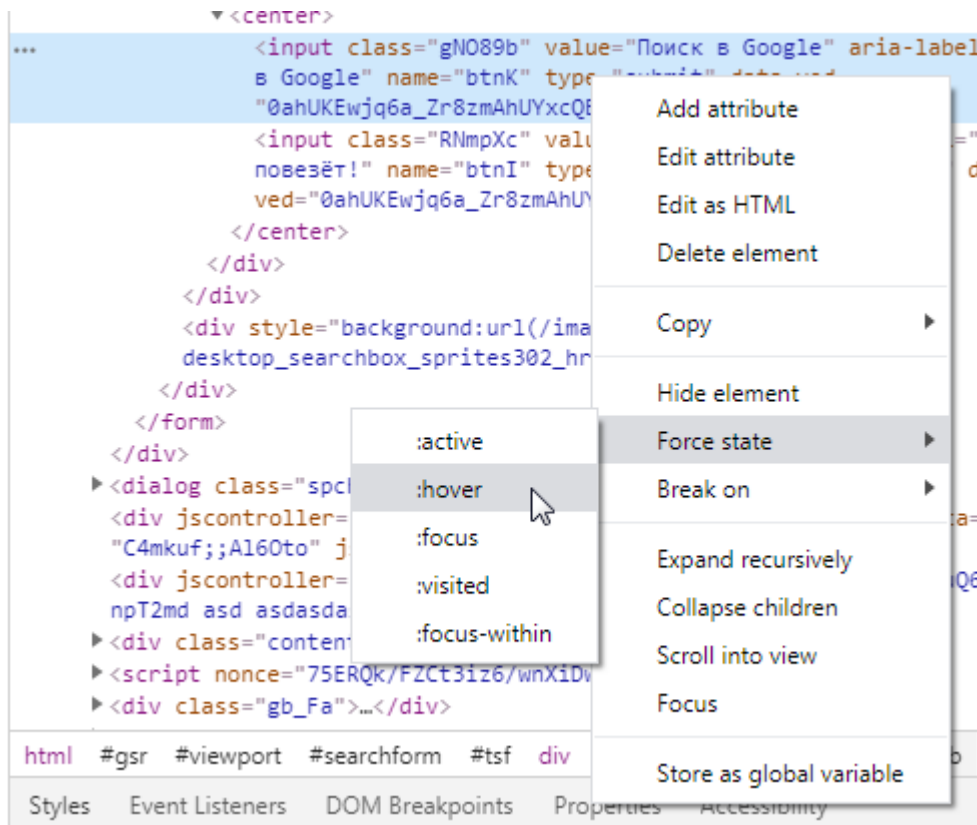


Через это меню можно быстро отредактировать элемент как HTML в текстовом поле для ввода, удалить элемент, спрятать его или добавить к нему атрибуты.

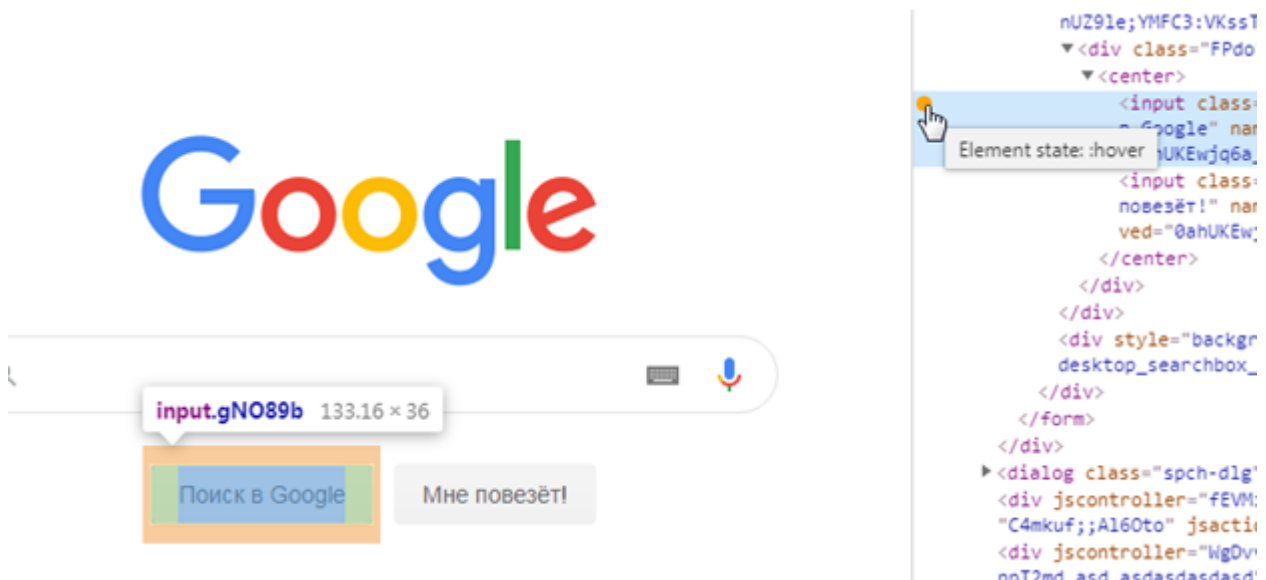
## Активация псевдоклассов

Псевдоклассы на элементах вызываются, чтобы исследовать, как элемент отреагирует, если на него, например, навести мышку. Мы можем активировать псевдоклассы `active`, `focus`, `hover` и `visited`.

Активировать псевдокласс можно через меню, доступное по правому клику на элемент:



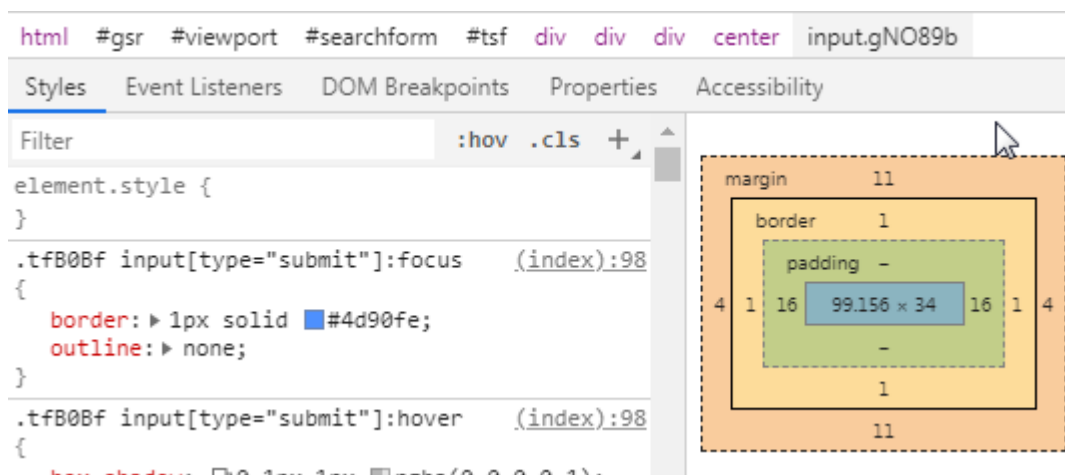
Когда к элементу применяется какое-то состояние, мы видим небольшой визуальный индикатор слева от открывающего тега, а в некоторых случаях, если они далеко друг от друга, и от закрывающего.



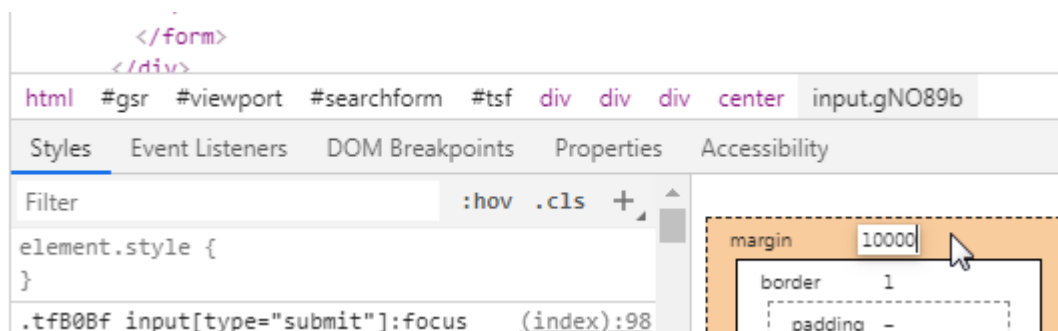
## Наглядное изменение стилей

Для каждого элемента в Elements открывается панель Styles, где можно менять, например, включать и выключать или редактировать, стили, цвета элементов, просматривать параметры внешних и

внутренних отступов:



Все параметры доступны для редактирования по двойному клику:



Но надо помнить, что эти изменения нигде не сохраняются. Всё, что мы отредактировали, потеряется при перезагрузке страницы.

## Console

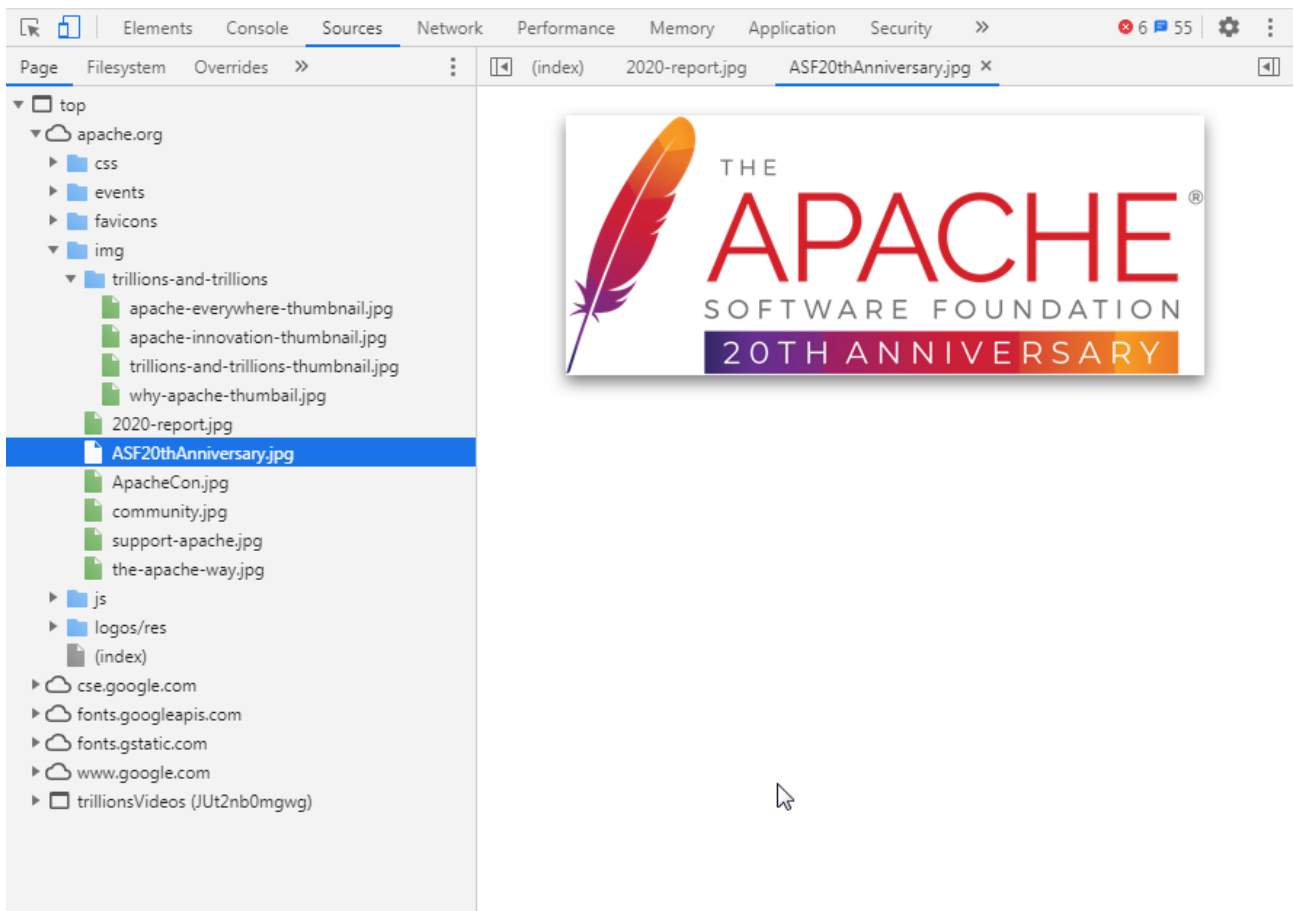
Вкладка Console позволяет просматривать, отлаживать и выполнять JS-код для загруженной страницы.

## Sources

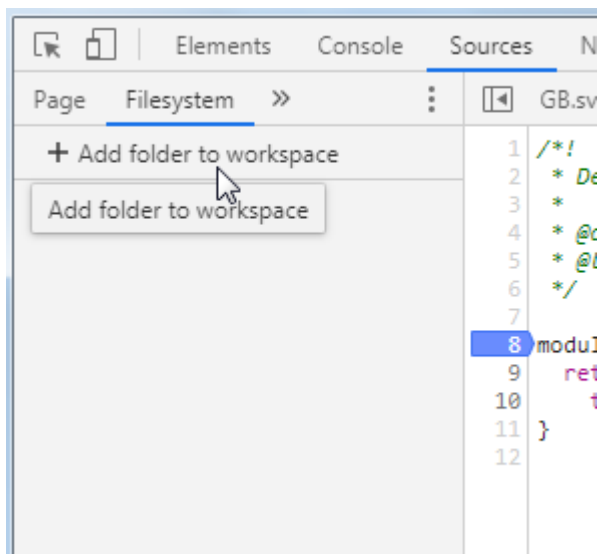
Основное назначение этой вкладки — работать с исходным кодом файлов веб-приложения.

Эта вкладка имеет несколько вложенных вкладок: Page, Filesystem, Overrides, Content Scripts, Snippets.

Вкладка **Page** показывает все файлы, которые используются на этой странице — сам исходный код, файлы стилей, код на JavaScript, изображения, шрифты и так далее. Так выглядит список файлов для сайта [Apache](#):



На вкладке Filesystem можно добавить папки с диска и посмотреть содержимое текстовых файлов (html, css, js), отредактировать код, скопировать его или сохранить изменённый файл как новый. Эта вкладка используется и как полноценный редактор кода, если подключиться к локальным файлам:





# Network

Эта вкладка позволяет мониторить процесс загрузки страницы и всех файлов, которые подгружаются при загрузке. Её удобно использовать для оптимизации загрузки страниц и мониторинга запросов.

На панели отображается таблица всех запросов к данным и файлам, над ней располагаются кнопки для фильтрации запросов, очистки таблицы или включения и отключения записи запросов, кнопки управления отображением таблицы. Есть также дополнительные переключатели:

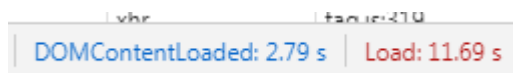
1. **Preserve log** — не очищать таблицу при перезагрузке страницы.
2. **Disable cache** — отключить кеш браузера — будет работать только при открытом DevTools.
3. **Offline** — эмулирует отсутствие интернета, также позволяет эмулировать скорость скачивания и загрузки данных и пинг для различных типов сетей.

Под таблицей указывается количество всех запросов, общее количество загруженных данных, общее время загрузки всех данных, время загрузки и построения DOM-дерева и время загрузки всех ресурсов, влияющих на отображение этой страницы.

## Ключевые возможности

1. Возможность отключить кеширование или установить ограничения пропускной способности.
2. Получение подробной таблицы с информацией о каждом запросе.
3. Фильтрация и поиск по всему списку запросов.

## Время загрузки страницы



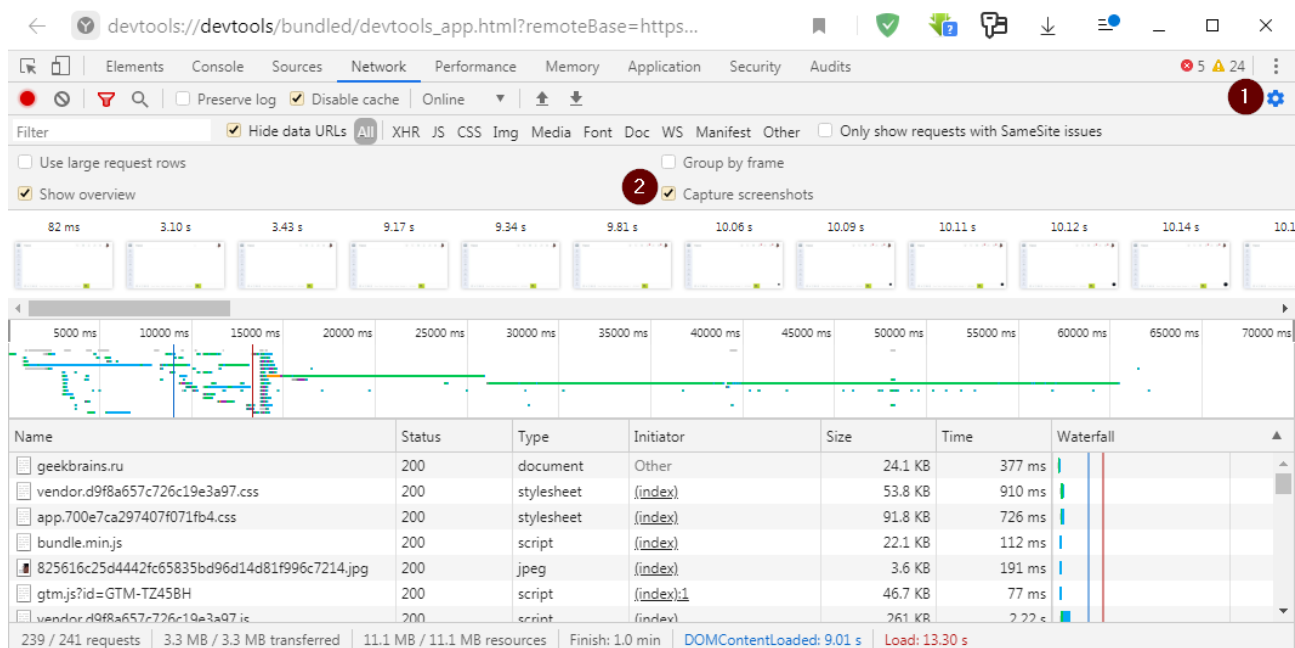
В инструментах разработчика можно увидеть время DOMContentLoaded и общее время загрузки. Чтобы его отобразить, выбираем панель Network и нажимаем Ctrl + R для обновления страницы. Появится синяя линия для DOMContentLoaded и красная — для общего времени загрузки. Обычно всё, что находится справа от синей линии или касается её, — это данные, которые блокируют DOM. Они называются ресурсами блокировки рендеринга.

206 B	395 ms		
206 B	344 ms		
636 B	343 ms		
707 B	304 ms		
30.5 KB	310 ms		
428 B	709 ms		
(disk cache)	206 ms		

## Режим диафильма

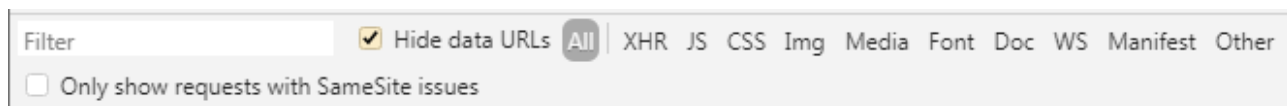
Функция захвата скриншотов позволяет видеть, как страница отображается от начала до конца. Она создаёт скриншоты в процессе загрузки. Это отличный способ увидеть, как рендерится страница в браузере при загрузке.

Для этого щёлкаем панель Network, открываем настройки, включаем Capture screenshots и нажимаем клавиши Ctrl + R для обновления страницы. Затем панель покажет, как страница отображается в процессе загрузки.



## Фильтры

Раздел фильтров позволяет показывать только элементы, удовлетворяющие определённым условиям.



В инструментах разработчика есть набор предустановленных фильтров для быстрой фильтрации элементов определённых типов:

1. All — все элементы, без фильтрации.
2. XHR (XMLHttpRequest) — запросы к серверу из JavaScript-кода.
3. JS — только файлы JavaScript (\*.js).
4. CSS — только файлы стилей (\*.css).

5. `Img` — только изображения: JPEG, GIF, PNG, SVG и другие.
6. `Media` — аудио- и видеоматериалы.
7. `Font` — шрифты.
8. `Doc` — документы, под которыми обычно подразумеваются сами HTML-страницы (`text/html`).
9. `WS` (`WebSocket`) — запросы по технологии веб-сокетов. Могут быть, например, в чатах.
10. `Manifest` — файлы манифеста. Специальные файлы для настройки отображения сайта на мобильных устройствах.
11. `Other` — остальное.

В поле `Filter` можно вводить специальные строки, чтобы отфильтровать результаты по конкретному запросу, например, `Is:running` — для просмотра всё ещё выполняющихся на странице запросов, `larger-than:1024` — для просмотра ресурсов размером более 1024 байт, `status-code:200` — для просмотра всех успешно загруженных ресурсов.

Можно использовать несколько свойств одновременно, разделяя их пробелами. Например, `mime-type:image/gif larger-than:1K` отображает все GIF-файлы, размер которых превышает один килобайт. Эти фильтры с несколькими свойствами эквивалентны операциям «и». Операции «или» в настоящее время не поддерживаются.

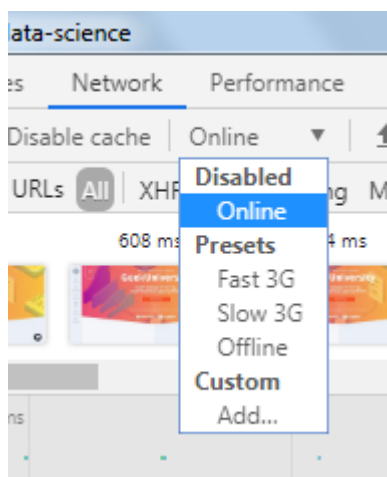
### Все поддерживаемые свойства

1. `domain:` — отображать только ресурсы из указанного домена. Маска (\*) используется для отображения нескольких доменов. Например, `*.com` отображает ресурсы со всех доменов, заканчивающихся на `.com`. DevTools автоматически подтягивает все домены, запрошенные сайтом, и добавляет их в подсказки.
2. `has-response-header:` — показать ресурсы, которые содержат указанный HTTP-заголовок в ответе.
3. `is:running` — позволяет найти ресурсы типа `WebSocket`.
4. `larger-than:` — показать ресурсы, размер которых превышает указанный размер в байтах. Установка значения 1 000 эквивалентна установке значения 1k.
5. `method:` — показать ресурсы, которые получены с использованием указанного HTTP-метода: `GET`, `POST` и так далее.
6. `mime-type:` — показать ресурсы с выбранным MIME-типом.
7. `mixed-content:` — показать все ресурсы со смешанным контентом (`mixed-content: all`) или только те, что отображаются сейчас (`mixed-content: display`).
8. `scheme:` — показать ресурсы, полученные по HTTP (`scheme:http`) или HTTPS (`scheme:https`).
9. `set-cookie-domain:` — показать ресурсы, которые имеют заголовок `Set-Cookie` с указанным доменом.
10. `set-cookie-name:` — показать ресурсы с заголовком `Set-Cookie` с именем, соответствующим указанному значению.

11. `set-cookie-value`: — показать ресурсы, которые имеют заголовок Set-Cookie со значением, соответствующее указанному в фильтре значению.
12. `status-code`: — отображать только те ресурсы, HTTP-код состояния которых соответствует указанному коду: 200, 301, 404 и так далее.

## Профили сети

Мы можем добавить настраиваемые профили сети. Это полезно, если надо протестировать свой проект на определённой скорости загрузки:



Чтобы добавить настраиваемый профиль, щёлкаем панель Network и выбираем раскрывающийся список Online. Затем кликаем на Custom — Add — Add custom profile и вводим название, скорость скачивания и загрузки, задержку.

### Network Throttling Profiles

[Add custom profile...](#)

Profile Name	Download	Upload	Latency
<input type="text"/>	kb/s	kb/s	ms
	optional	optional	optional

[Add](#) [Cancel](#)

## Performance

Панель отображает таймлайн использования сети, выполнения JavaScript-кода и загрузки памяти. После первоначального построения графиков таймлайна станут доступны подробные данные о выполнении кода и обо всём жизненном цикле страницы. Можно будет ознакомиться со временем исполнения отдельных частей кода, появится возможность выбрать отдельный промежуток на временной шкале и ознакомиться с тем, какие процессы происходили в этот момент.

Инструмент применяется для улучшения производительности страницы в целом.

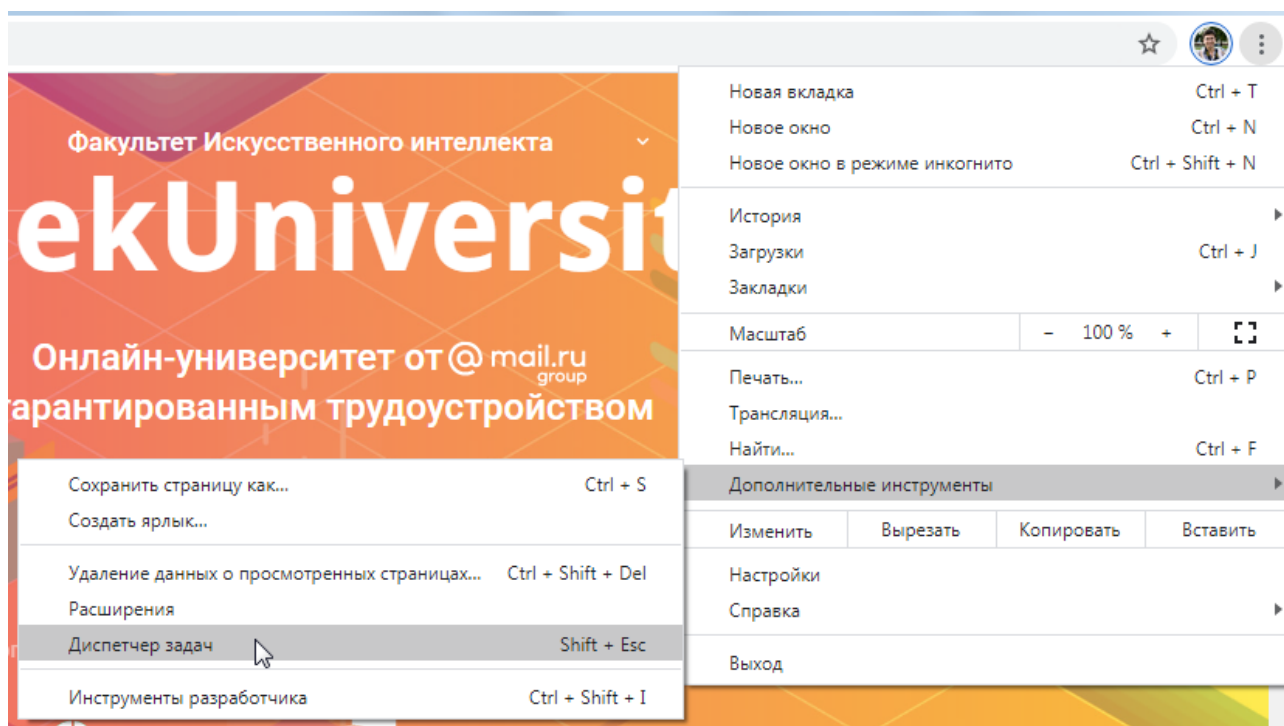
## Ключевые возможности

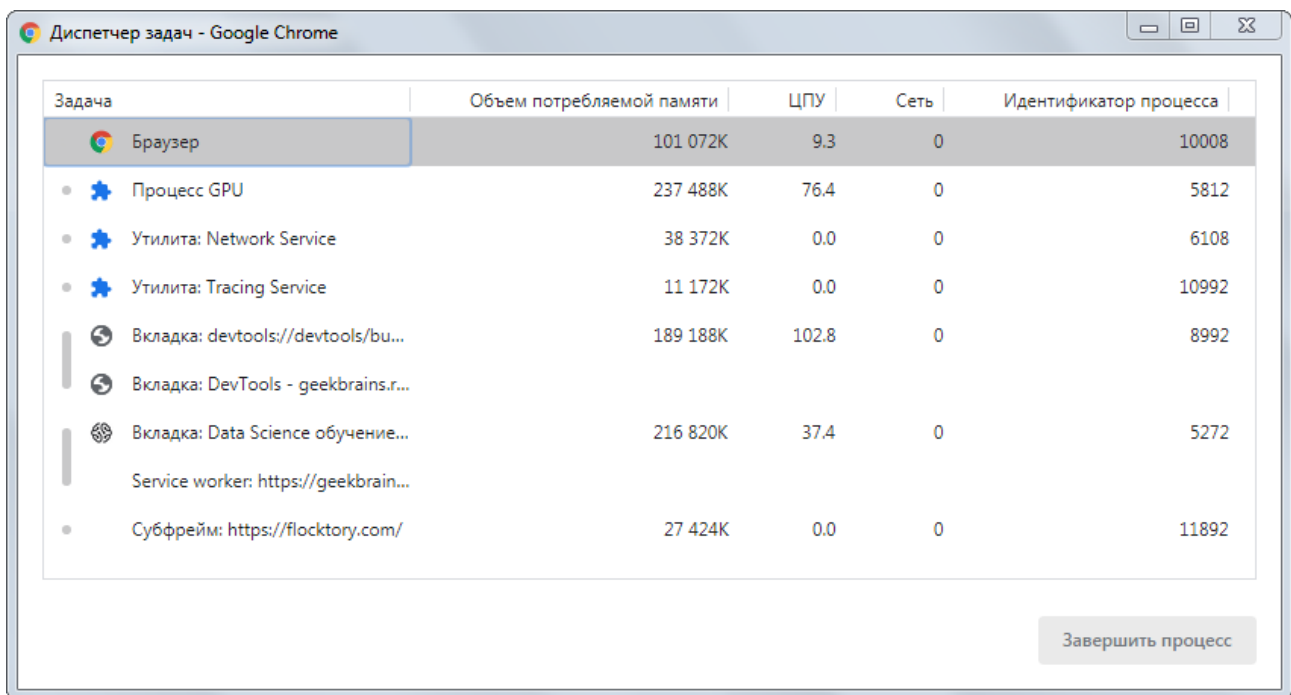
1. Возможность сделать запись, чтобы проанализировать каждое событие, которое произошло после загрузки страницы или взаимодействия с пользователем.
2. Возможность просмотреть FPS, загрузку CPU и сетевые запросы в области Overview.
3. Возможность посмотреть детали по каждому событию.
4. Возможность изменить масштаб таймлайна, чтобы сделать анализ проще.

# Memory

Панель «Память» в Chrome Developer Tools предоставляет информацию о том, как страница использует память. Утечки памяти происходят, когда веб-сайт потребляет больше ресурсов, чем необходимо. Серьёзные утечки памяти могут даже сделать сайты непригодными для использования.

Чтобы узнать, сколько памяти потребляют разные страницы, надо открыть встроенный диспетчер задач Chrome.





Открыв вкладку Chrome Memory, мы познакомимся с тремя вариантами исследования по использованию памяти веб-сайта:

1. **Heap snapshot** — распределение памяти по DOM.
2. **Allocation instrumentation on timeline** — выделение памяти в процессе загрузки страницы.
3. **Allocation sampling** — измерение выделенной памяти для «долгоиграющих» запросов.

#### Ключевые возможности

1. [Исправление проблем с памятью.](#)
2. [Профилирование CPU при работе с JavaScript.](#)

## Application

Вкладка для инспектирования и очистки всех загруженных ресурсов, в том числе IndexedDB или Web SQL баз данных, local и session storage, cookie, кеша приложения, изображений, шрифтов и таблиц стилей.

#### Ключевые возможности

1. Быстрая очистка хранилищ и кеша.
2. Инспектирование хранилищ, баз данных и кеша, управление ими.
3. Инспектирование и удаление файлов cookie.

## Security

На вкладке можно ознакомиться с протоколом безопасности при его наличии и просмотреть данные о сертификате безопасности, если он есть.

Инструмент используется для отладки проблем смешанного контента, проблем сертификатов и так далее.

#### Ключевые возможности

1. Окно Security Overview быстро подскажет, безопасна текущая страница или нет.
2. Есть возможность ознакомиться с отдельными источниками, чтобы:
  - просмотреть соединение и детали сертификата (для безопасных источников);
  - или узнать, какие запросы не защищаются (для небезопасных источников).

## Lighthouse

После выбора настроек и нажатия кнопки Run панель аудита анализирует, как загружается страница, и затем предоставляет предложения по оптимизации для уменьшения времени загрузки страницы и увеличения её отзывчивости.

Анализируются такие параметры: кеширование ресурсов, gzip-сжатие, наличие неиспользуемых частей JS-кода и CSS-правил и многое другое. Далее пользователю выводится сгруппированный список рекомендаций, благодаря выполнению которых можно существенно оптимизировать скорость загрузки и отзывчивость страницы.

## Практическое задание

### Задание 1

Проанализируйте сайт через Chrome Developer Tools. Сайт можно взять любой: над которым вы работаете, сайт вашего работодателя и так далее. Если у вас нет сайта, то выберите, например, из следующих:

1. <https://apache.org/> — лёгкий уровень сложности. Сайт простой, содержит минимум разных ресурсов, с хорошей и читаемой вёрсткой.
2. <https://gb.ru> — средний уровень сложности, но мы его разбирали на лекции, так что сложности не возникнут.
3. <https://mail.ru> или <https://yandex.ru> — средний уровень сложности.
4. <https://mail.google.com> — если у вас есть аккаунт, почта в Google или что-то аналогичное. Повышенный уровень сложности.

#### Что надо сделать:

1. Elements: отредактировать DOM, добавить/удалить элементы. Отредактировать CSS-свойства.
2. Console: выписать ошибки (Errors) и/или предупреждения (Warnings), зависит от количества найденного. Одну из ошибок, если они есть, описать как баг — заголовок, окружение, шаги, ОР, ФР и прочее.
3. Sources: определить, какие ресурсы загружаются сайтом. Сделать prettyprint для одного из CSS/HTML/JS-файлов (значок { }).
4. Network:
  - a. Определить время загрузки страницы.
  - b. Выяснить, какие элементы грузятся дольше всего.
  - c. Для самой страницы (HTML-файла) выяснить HTTP-заголовки запроса и ответа.
  - d. Узнать, сколько времени клиент ждал ответа — параметр Waiting (TTFB).
  - e. Посмотреть загрузку сайта на медленном интернете.
  - f. Сделать скриншоты и посмотреть процесс загрузки.
5. Performance: запустить профайлинг и затем посмотреть в Summary, на что потрачено время.
6. Memory: сделать Heap Snapshot и посмотреть, сколько памяти заняла страница. Можно переключиться с Summary на Statistics.
7. Application: проверить и «почистить» cookie.
8. Security: проверить, всё ли подключено через HTTPS, нет ли проблем с сертификатами.

Формат сдачи: набор скриншотов и описания багов в формате PDF в одном файле.

#### Задание 2. Тест для самопроверки

<https://coreapp.ai/app/player/lesson/614a22ce8478af554b6f4dc3> (сдавать не нужно)

## Глоссарий

**Chrome DevTools** — инструменты для разработки и тестирования веб-приложений.



# Дополнительные материалы

1. Статья [Pause your code with breakpoints.](#)
2. [Updates.](#)

# Используемые источники

1. Статья [«Обзор всех инструментов разработчика Chrome DevTools».](#)
2. [Google Chrome DevTools.](#)