Тестування програмного забезпечення

Викладач: к.т.н. Шитікова Олена Вікторівна

Лекція 3
ВЕБТЕСТУВАННЯ ТА ЧЕК-ЛИСТИ

Поняття вебтестування та основні підходи

Компанії та окремі користувачі все більше залежать в своїй роботі від вебдодатків (web-applications), які дуже динамічні, а їх функціональні можливості постійно зростають.

Безперервно зростає і потоковий трафік засобів інформації і запитів, які формуються переносними та вбудованими пристроями. Вебдодатки стають все більш поширеними і більш складними, граючи, таким чином, основну роль в більшості онлайнових проєктів.

Як і у всіх системах, що ґрунтуються на взаємодії між клієнтом і сервером, вразливості вебдодатків зазвичай виникають із-за некоректної обробки запитів клієнта чи недостатньої перевірки вхідної інформації зі сторони розробника.

Поняття вебтестування та основні підходи

Важливою особливістю вебтестування є часові рамки. Якщо при тестуванні програмного забезпечення в тестувальника є місяці (інколи роки), то у вебтестувальника є лише дні і тижні (в кращому випадку) на тестування запропонованого сайту.

Отже, якщо при тестуванні програм дозволяється і потрібно формувати детальні плани тестування, описувати тестові випадки на основі отриманої від розробника документації, то при тестуванні вебсторінок це може значно продовжити строки публікації матеріалів в мережі Інтернет. Таким чином, на вебтестування може виділятися до 50% загального бюджету і часових ресурсів.

Поняття вебтестування та основні підходи

Вебтестування (Web-testing) – тестування програмного забезпечення, сфокусоване на вебдодатках (web-applications), яке здійснюється з застосуванням систем, що використовують вебтехнології (web-basedsystem), та вирішують такі питання як: безпека вебдодатків, базова функціональність сайту, доступність користувачам тощо.

Поняття вебтестування та основні підходи

Вебдодатки (Web-applications) – клієнт серверний додаток, в якому клієнтом виступає браузер, а сервером - вебсервер.

Вебсайт (Website) – система з декількох сторінок, що мають одну адресу в мережі Інтернет.

Верстка (Layout) – етап дизайну сторінки сайту, що представляє собою просторове розміщення на ній текстових елементів і графічних зображень відповідно до концепції оформлення ресурсу.

Вебдизайн (Web-design) – галузь веброзробки, а також різновид дизайну, до завдань якої входить проектування вебінтерфейсів (Web UI) для сайтів або вебдодатків.

Етапи тестування (Stages of testing)

1. Вивчення документації (Documentation) – вивчається отримана документація, аналізується функціонал за технічним завданням, кінцеві макети сайту і складається план тесту для подальшого тестування. Необхідно визначити цілі сайту і споживачів сайту.

2. Тестування верстки (Layout testing) – перевіряється розташування елементів, відповідність їх позицій наданим макетам, а також перевіряється оптимізація зображень і графіки. Завершивши перевірку на валідність, фахівець приступає до перевірки на кроссбраузерність, тобто перевіряє працездатність сайту в різних браузерах та при різних параметрах налаштувань екрану.

Етапи тестування (Stages of testing)

3. Функціональне тестування (Functional testing) – процес верифікації відповідності функціонування продукту його початковим специфікаціям і представляє найбільш тривалий етап перевірки ресурсу.

Перевіряється весь описаний функціонал:

- посилання;
- бази даних;
- кешування;
- фрейми;
- аудіо і відео;
- форми;
- секретність;
- перевірка роботи з браузером;
- анімація;
- друк;
- включення або відключення куків (cookie).

Етапи тестування (Stages of testing)

4. Тестування практичності / простоти використання (Usability testing) – проводиться для оцінки зручності продукту у використанні:

– *тестування навігації* (Test for navigation) - вебсайт повинен бути простим у використанні, інструкції повинні бути зрозумілими і головне меню повинно бути передбаченим на кожній сторінці і бути послідовним;

– *перевірка вмісту* (Content checking) – зміст має бути логічним і простим для розуміння без орфографічних помилок, без використання кольорової гами, яка дратує користувачів.

Етапи тестування (Stages of testing)

Тестування практичності одне з найдорожчих, тому що найбільш цінну інформацію можна отримати тільки від реальних користувачів, спостерігаючи за їх роботою з сайтом. Такі дослідження вимагають дорогої інфраструктури і тимчасових витрат, їх складно автоматизувати. Тестування практичності / простоти використання (usability testing) частково перетинається з тестуванням якості виконання сайту і зручності інтерфейсу.

Етапи тестування (Stages of testing)

5. Тестування безпеки (Security testing) – перевіряється доступ користувача до службових/закритих сторінок, а також проводиться перевірка захисту всіх критично важливих сторінок (наприклад, розділу адміністрування сайту) від зовнішнього впливу і включає наступні дії:

- виявлення помилок в коді сайту та програмному забезпеченні сервера;
- виявлення наявності таких вразливостей, як SQL Injection або ін'єкція XSS (Cross SiteScripting), PHP includes, биті посилання (broken links);
- тестування на стійкість до комбінованих методів атак і нестандартних технік зламування.

Етапи тестування (Stages of testing)

На відміну від інших тестів, тестування безпеки слід проводити регулярно. Крім того, тестуванню піддається не тільки сам сайт або вебдодаток, а весь сервер повністю – і вебсервер, і операційна система, і всі мережеві сервіси. Як і у випадку інших тестів, програма «прикидається» реальним користувачем-хакером і намагається застосувати до сервера всі відомі методи атаки і перевіряє всі вразливості. Результатом роботи буде звіт про знайдені вразливості та рекомендації щодо їх усунення. Зазвичай такі сканери безпеки продаються як самостійний продукт.

Етапи тестування (Stages of testing)

6. Тестування продуктивності сайту (Performance testing) Проводиться з метою визначення швидкодії сайту або частини під певним навантаженням.

Тестування продуктивності включає в себе наступні види тестування продуктивності:

- Навантажувальне тестування;
- Тестування швидкодії

Етапи тестування (Stages of testing)

- ***Навантажувальне тестування*** (Load-testing) – найпростіша форма тестування продуктивності.

Тестування навантаження зазвичай проводиться для того, щоб оцінити поведінку сайту (або програми) під заданим очікуваним навантаженням. Цим навантаженням може бути, наприклад, очікувана кількість одночасно працюючих користувачів на сайті, що виконують задане число транзакцій за інтервал часу. Такий тип тестування зазвичай дозволяє отримати час відгуку всіх найважливіших бізнес-функцій.

Етапи тестування (Stages of testing)

Основні цілі навантажувального тестування:

- оцінка продуктивності і працездатності програми на етапі розробки і передачі в експлуатацію;
- оцінка продуктивності і працездатності програми на етапі випуску нових релізів, інформації, призначеної для автоматизованого внесення певних змін в комп'ютерні файли (patch);
- оптимізація продуктивності додатку, включаючи налаштування серверів та оптимізацію коду;
- підбір відповідної для цього додатку апаратної (програмної платформи) і конфігурації сервера.

Етапи тестування (Stages of testing)

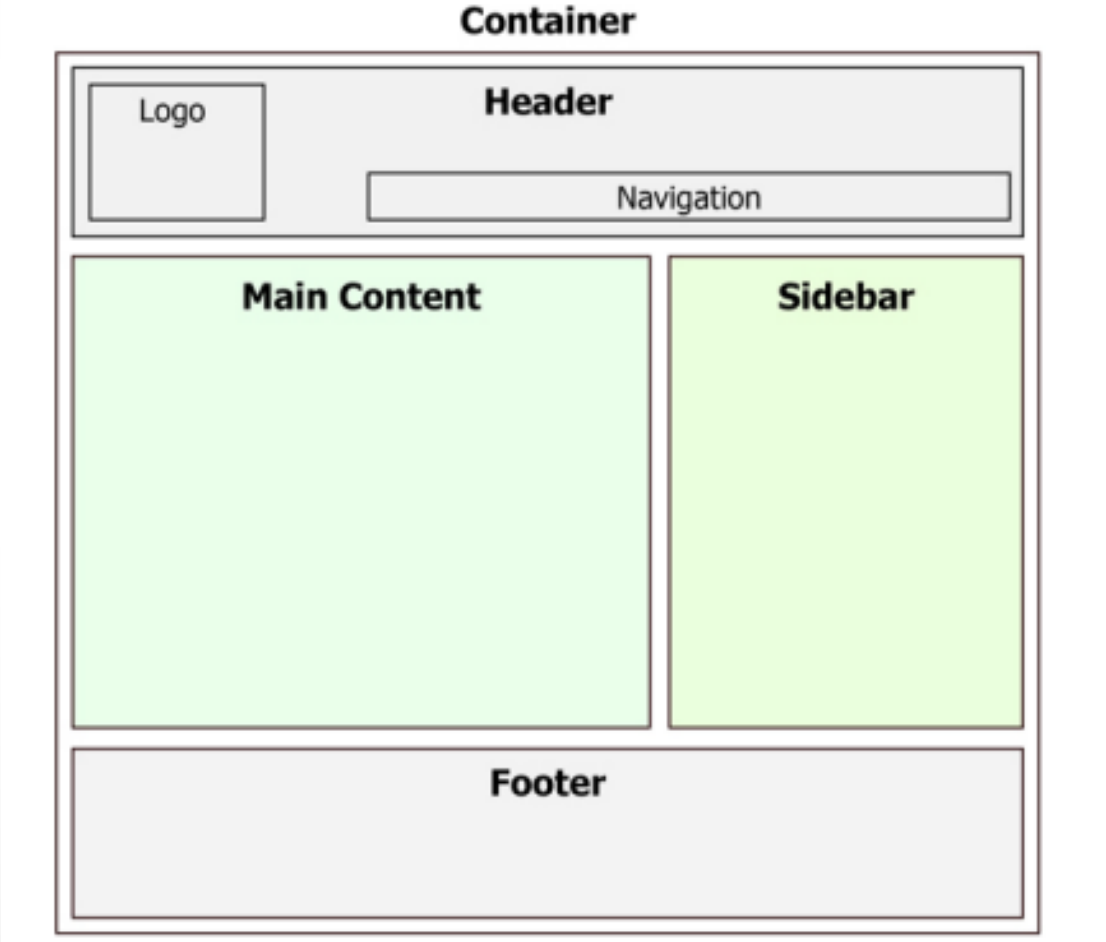
- ***Тестування швидкодії*** – перевірка швидкості завантаження сайту для визначення швидкості відпрацювання скриптів, завантаження зображень і контенту.

Цей тест проводиться з метою оптимізації процесу завантаження сайту, а також визначення оптимальності налаштувань сервера.

У такому тесті перевіряється не тільки і не стільки сам сайт, скільки спільна злагоджена робота всього комплексу – апаратної частини сервера, вебсерверу, програмного ядра (engine) та інших компонентів сайту.

Анатомія вебсторінки

Елементи вебсторінки забезпечують життєво важливу і естетичну функцію сторінки.



Тестування верстки

1. Візуальна частина. Тестування проводиться в одному з найчастіше використовуваному браузері в наступній послідовності:

- перевіряються помітні оку поламані блоки, невідповідність кольору, некоректне відображення тексту навколо зображень;
- точність відповідності макету;
- перевіряється сітка (вертикальні/горизонтальні вирівнювання).
- сайт перевіряється в різних розширеннях.

Тестування верстки

- перевіряється зменшення розмірів вікна менше мінімального по технічному завданню – не повинно нічого «ламатися», фони не повинні «розпливатись»;
- перевіряється масштабування сторінки. У розумних масштабах (діапазон 75-150%) сторінка повинна виглядати без візуальних неточностей;
- перевіряється чи нормально підсвічуються поля у фокусі;
- перевіряється чи нормально підсвічуються поля з помилками.

Тестування верстки

2. Доступність включає наступні етапи:

• чи виділяється текст в текстових блоках;

• чи натискаються «клікабельні» елементи;

• чи встановлюється фокус в поля форм;

• в ідеальному випадку всі активні елементи повинні реагувати на наведення, не доступні/неактивні – не повинні;

• клікабельні елементи, призначення яких не очевидно, повинні бути забезпечені підказками (title);

Тестування верстки

- лого на головній сторінці посиланням бути не повинно, на всіх внутрішніх сторінках – повинно;
- перевірка друку сторінки (якщо було в ТЗ);
- написи (особливо логотип і головне меню сайту) повинні залишатися читабельними, для всіх інформаційних картинок повинні бути присутні підписи;
- працездатність при вимкненому JavaScript. Весь критично важливий функціонал сайту повинен бути доступний без JS.

Тестування верстки

3. Коректна робота при введенні реального тексту, надійність верстки.

Верстка повинна «тягнутися», не розвалюватися та не втрачати дизайнерський вигляд при зміні контенту на сторінці. Контенту може бути більше або менше, ніж на макеті:

– Перевірка введення та видалення даних.

– Перевірка коректності роботи стилів.

Тестування верстки

4. Перевірка 404-их запитів. Помилка 404 зобов'язана вписуватися в загальну концепцію сайту, мати ідентичний стиль, надавати відвідувачам можливість повідомляти адміністратора щодо непрацюючих URL.

404 сторінка зобов'язана вирішувати такі ***завдання***:

- привертати увагу,
- пояснювати користувачам ситуацію,
- підказувати можливі варіанти подальших дій.

Тестування верстки

Щоб сторінка 404 приносила ресурсу користь, вона повинна включати наступні елементи:

- перехід на головну сторінку;
- кнопка повідомлення про бите посилання;
- список популярних матеріалів;
- форму пошуку;
- посилання на карту сайту.

Чек-листи

Чек-лист – це документ, який описує що має бути протестовано. При цьому чек-лист може бути абсолютно різного рівня деталізації. На скільки детальним буде чек-лист залежить від вимог до звітності, рівня знання продукту співробітниками й складності продукту.

Чек-лист дозволяє:

- не забувати про важливі тести;
- фіксувати результати роботи;
- відстежувати статистику про статус ПЗ.

Чек-листи

В чек-листі має бути:

✔ список перевірок (з необхідним ступенем деталізації);

✔ оточення перевірки:

- збірка, на якій проводилося тестування;
- тестове оточення (якщо є);
- інформація про тестувальників;

✔ результат перевірки.

Чек-листи

Приклад чек-листа

	Тестувальник	*Ольга Д.*	*Анна П.*	*Анна П.*	*Ольга Д.*	
	Перевірки	***Google Chrome 68.0.3285.140***	***Mozilla Firefox 58.0.2***	***Opera 51.0.28.30.34***	***Safari 5.1.7***	***ID баг-репорту***
1	Наявність сторінки логіна	Passed	Passed	Passed	Passed	-
2	Авторизація з валідними даними	Passed	Passed	Passed	Passed	-
3	Авторизація з невалідними даними	Failed	Failed	Failed	Failed	005475
4	Авторизація з порожніми полями	Failed	Failed	Failed	Failed	005477
5	Реєстрація з валідними даними	Passed	Passed	Failed	Passed	005485
6	Реєстрація з невалідними даними	Passed	Failed	Failed	Passed	005493
7	Лист підтвердження реєстрації	Blocked	Blocked	Blocked	Blocked	005495

Чек-листи

Пункти можуть містити як лінійну структуру, так і деревоподібну, з розділами/підрозділами або без них. Вони повинні бути максимально короткі та в той же час зрозумілі тестувальнику, який ще не знайомий з додатком. Всі пункти повинні бути оформлені однією мовою.

Як правило, кожен чек-лист має кілька стовпців. Кожен стовпець призначений для тестування на окремій платформі. Слід завжди вказувати назву пристрою, назви браузерів та їх версії.

Чек-листи

При проходженні чек-листів тестувальник зазначає статус навпроти кожного тестованого пункту. Можливі такі варіанти статусів:

- «Passed» – перевірка пройдена успішно, багів не знайдено;
- «Failed» – знайдений один або більше багів;
- «Blocked» – неможливо перевірити, тому що один з багів блокує поточну перевірку;
- «In Progress» – поточний пункт, над яким працює тестувальник;
- «Not run» – ще не перевірено;
- «Skipped» – не буде перевірятися з будь-якої причини. Наприклад, поточний функціонал ще не реалізований.

Чек-листи

Для більшої наочності, як правило, кожен зі статусів має свій колір.

Всі заведені по чек-листу баг-репорти повинні бути додані в примітки до комірок зі статусом «Failed». Цілком припустимо додавати примітки до комірок з іншими статусами, якщо це необхідно.

Для комірок зі статусом «Blocked» примітки з посиланнями на баг-репорти також обов'язкові. Однак, як правило, примітка в комірці зі статусом «Blocked» посилається на раніше заведений баг-репорт, який в одному з попередніх пунктів вже відзначений як «Failed». Іншими словами: пункт, одного разу зазначений як «Failed», може бути блокером для декількох або всіх наступних пунктів чек-листа

Чек-листи

Основні моменти, які варто враховувати при роботі з чек-листами:

✔ статус «Passed» встановлюється тільки для пунктів, які перевірені і не містять помилок;

✔ всі комірки зі статусом «Failed» і «Blocked» обов'язково повинні мати примітки з посиланнями на баг-репорти;

✔ після завершенні проходження чек-листа не повинно залишитися комірок зі статусом «Not run».

Чек-листи

Правила складання чек-листів:

1. ***Один пункт – одна операція.*** Пункти чек-листа – це однозначні атомарні і повні операції.
2. ***Починати пункти з іменника.*** Складаючи пункти слід дотримуватися уніфікованої форми.
3. ***Складати чек-лист за рівнями деталізації.*** Для зручності проходження чек-листа, найкраще складати тести в послідовності використання функціоналу.

Чек-листи

Переваги чек-листів:

✔ Використання чек-листів сприяє структуруванню інформації у співробітника.

✔ Чек-лист стає пам’яткою, яка допомагає не забути важливі тести.

✔ При правильному запису необхідних дій у співробітника з'являється однозначне розуміння завдань. Це сприяє підвищенню швидкості навчання нових співробітників.

Чек-листи

Переваги чек-листів:

✔ Чек-листи допомагають уникнути невизначеності та помилок пов’язаних з людським фактором. Збільшується покриття тестами програмного продукту.

✔ Підвищується ступінь взаємозамінності співробітників.

✔ Економія робочого часу. Написавши чек-лист одного разу його можна використовувати повторно, з огляду на актуальність інформації.

Чек-листи

Переваги чек-листів:

✔ Використання статистики: хто, коли, що проходив (з деталізацією по збірці продукту та оточенню, на якому проводилося тестування).

✔ Можливість оцінити стан продукту, готовність до випуску.

Чек-листи

Недоліки чек-листів:

✔ початківці тестувальники не завжди ефективно проводять тести без достатньо докладної документації;

✔ чек-листи неможливо використовувати для навчання початківців співробітників, так як в них недостатньо докладної інформації;

✔ замовнику або керівництву може бути недостатньо того рівня деталізації, який пропонують чек-листи.

Інструменти розробника (Developer Tools)

Інструменти розробника – це потужний інструмент для налагодження коду вебсайтів, який за замовчуванням встановлений в багатьох браузерах (Firefox, Google Chrome, Edge та в інших).

Крім веброзробки, даний інструментарій буде дуже корисний тестувальникам при тестуванні вебайтів і створенні баг-репортів.

Інструменти розробника (Developer Tools)

Для того, щоб в браузері **Google Chrome** відкрити вкладку Console можна:

- скористатися гарячими клавішами *Command - Option - I* (Mac) або *Control - Shift - I* (Windows/Linux);
- вибрати пункт меню *Додаткові інструменти* > *Інструменти розробника* > *Console*;
- натиснути клавішу F12.

Ці способи Developer Tools також працюють і в інших браузерах, таких як **Mozilla Firefox**, **Edge**.

Інструменти розробника (Developer Tools)

Для того, щоб відкрити консоль в браузері **Safari**, необхідно в налаштуваннях браузера активувати меню розробника (якщо воно відсутнє), натиснувши

Safari > Preferences > Advanced, і відмітити *Show Develop menu in menu bar,* після цього визвати консоль за допомогою поєднання клавіш

Command+Shift+I або *Develop > Show Web Inspector*.

Інструменти розробника (Developer Tools)

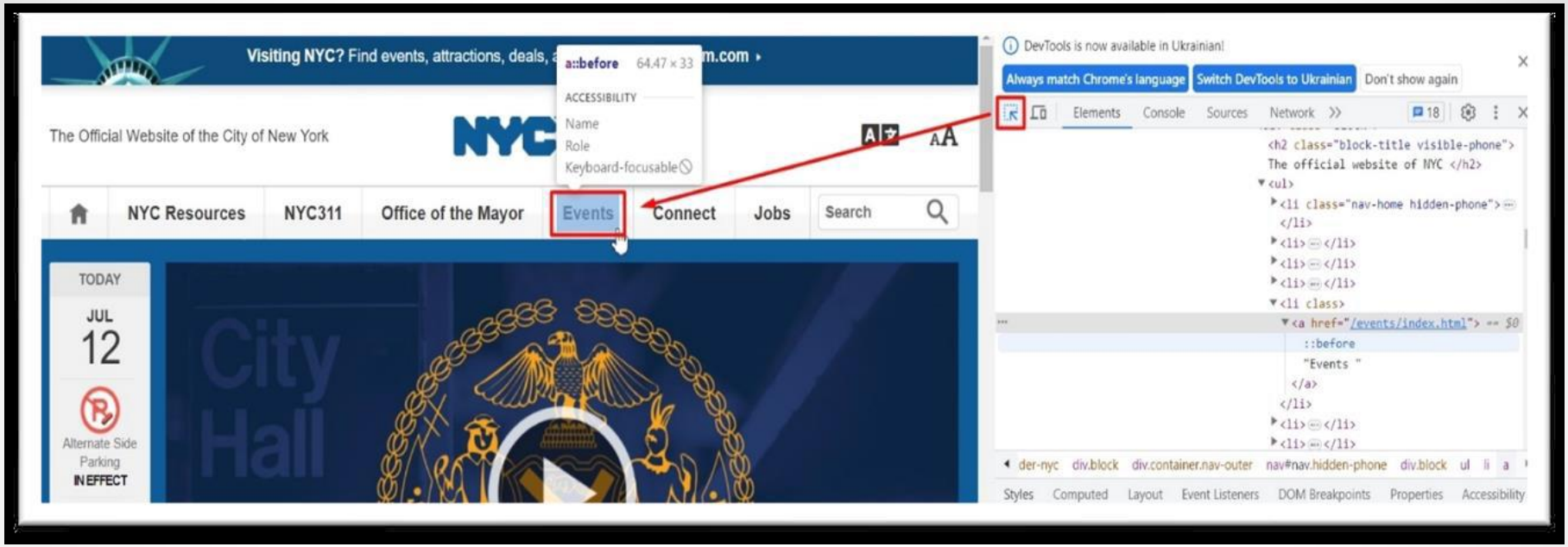
Функції Developer Tools:

- перегляд HTML та CSS-коду елементів вебсторінки (вкладка «Elements»);
- емуляція відображення вебсторінки на мобільному пристрої (вкладка-іконка «Toggle device Toolbar»);
- моніторинг подій на сторінці з відображенням попереджень і помилок (вкладка «Console»);
- виконання JavaScript-команд (вкладка «Console»);
- перегляд всіх ресурсів, що завантажуються на сторінці: картинок, скриптів, таблиць стилів, також можна правити скрипти і можна завантажити на ПК файли (вкладка «Sources»);
- перегляд статистики завантаження елементів, а також запитів, що відправляються сторінкою (вкладка «Network»);

Інструменти розробника (Developer Tools)

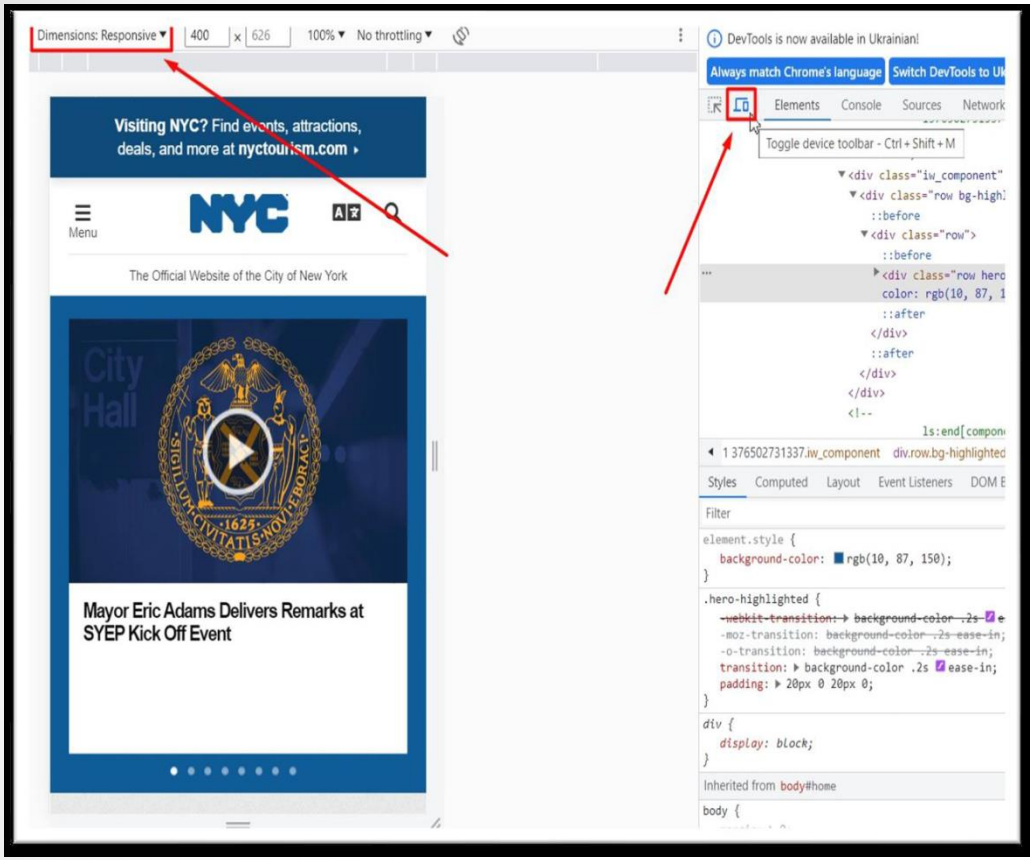
- емуляція завантаження сторінки із штучним обмеженням швидкості (вкладка «Network»);
- оцінка навантаження вебресурса на систему користувача для оптимізації (вкладка «Performance»);
- перегляд поточного використання пам’яті вебсторінкою (вкладка «Memory», наявна не у всіх версіях браузерів);
- перегляд та редагування даних сесій, cookie-файлів та інших локальних сховищ вебсторінки (вкладка «Application»);
- перегляд звіту про безпечність ресурсів (вкладка «Security»);
- перегляд рекомендацій для оптимізації сторінок (вкладка «Audits»).

1. Дослідження елементів



Крайня ліва кнопка в інспекторі активує інструмент для виділення елементів. Якщо навести вказівник миші на будь-який елемент, можна дізнатися його назву чи ID, розміри, внутрішні і зовнішні відступи, межу. Якщо клікнути на виділений елемент, у вікні нижче можна побачити його виділеним всередині структури DOM

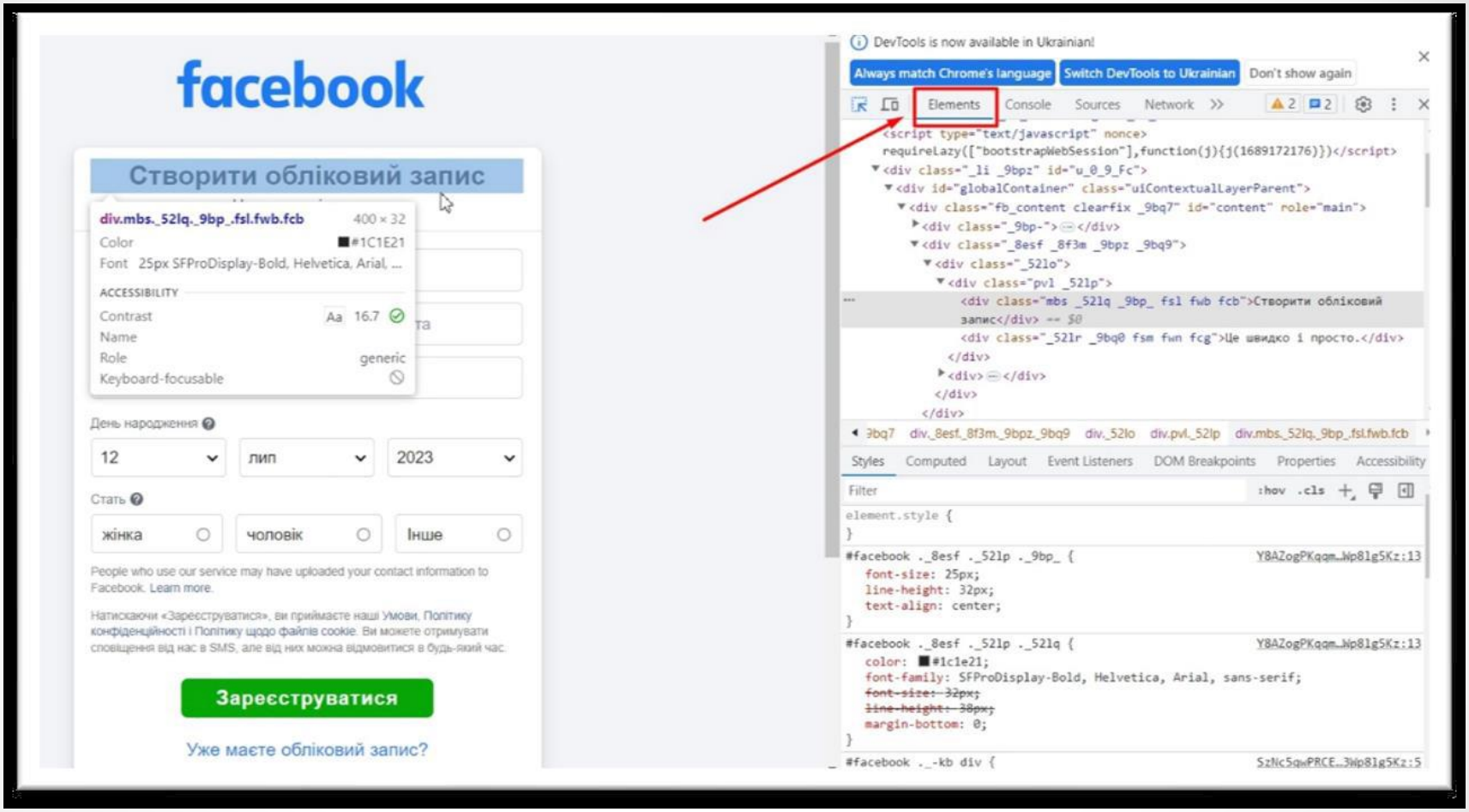
2. Адаптивний дизайн



Наступна кнопка (Toggle device Toolbar) дозволяє вибрати режим емуляції мобільного пристрою. Це зручний інструмент для тестування в різних розширеннях екрану і перевірки відображення сайту на мобільних пристроях.

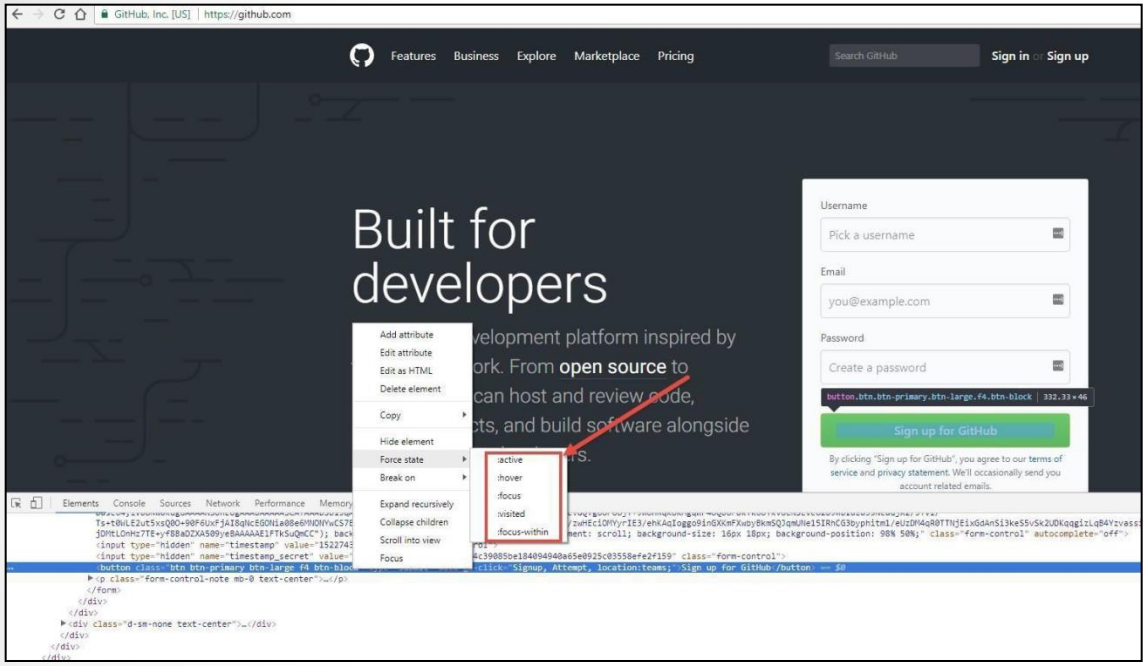
У верхньому меню можна вибрати тип пристрою для емуляції, встановити розширення екрану вручну, змінити орієнтацію з вертикальної на горизонтальну та навпаки.

3. Elements (Елементи)



Використовується для перегляду коду сторінки, відображається у вигляді DOM – дерева елементів.

3. Elements (Елементи)



Після кліку правою клавішею миші на будь-який елемент можна додати атрибути, редагувати їх, видалити елемент із структури сторінки або приховати його, чи додати елементу псевдокласи.

Для зміни HTML-елемента необхідно вибрати потрібний тег, натиснути F2 і в блоці, що відкриється, можна буде змінити його зміст, додати інші атрибути цьому тегу, а також додати інший тег перед чи після даного.

4. Console (Консоль)

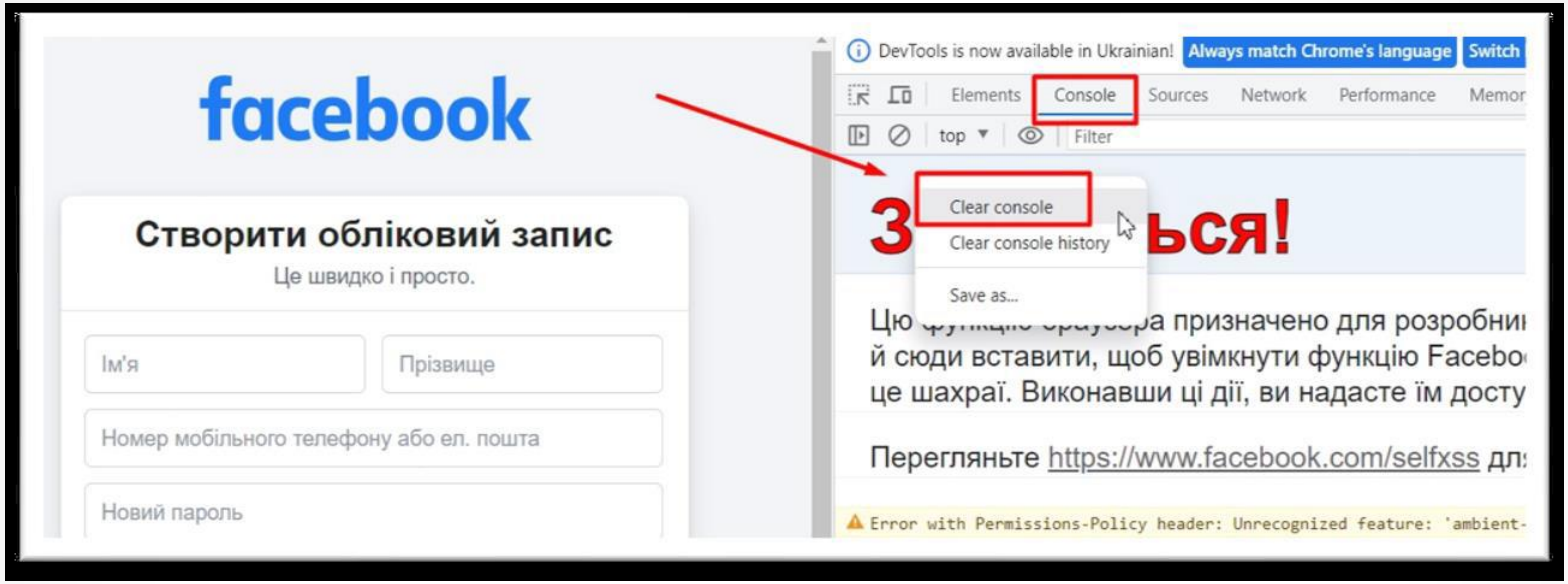
В цій вкладці відображаються важливі повідомлення про роботу сайта, помилки і попередження, логи для розробників, а також в консолі можна виконувати JavaScript код.

Очищення консолі:

Для очищення консолі необхідно:

- Клікнути правою клавішею миші і в контекстному меню вибрати пункт Clear Console.
- Ввести команду clear() з Command Line API в консолі.
- Визвати команду console.clear() із Console API із скрипта.
- Використати гарячі клавіші: ⌘K або ^L (Mac) Control – L (Windows і Linux).

4. Console (Консоль)



Якщо клікнути правою клавішею миші, то відкриється контекстне меню, в якому можна очистити консоль (Clear Console), очистити історію введення команд і параметрів в консоль (Clear Console History), зберегти лог консолі в окремий файл.

4. Console (Консоль)

Необхідно звернути увагу на те, що логи зберігаються в рамках поточної сторінки з моменту відкриття консолі, а не завантаження сторінки. Тому, якщо спочатку відкрити сторінку, а потім відкрити консоль, то помилки не будуть відображатися, хоч і будуть на сторінці.

Також варто звернути увагу на те, що консоль відображає також помилки різних встановлених плагінів і розширень для браузера, враховуючи віруси (вони також прописуються як плагіни або розширення).

5. Sources

Вкладка, в якій відображаються всі файли, що підключені до поточної сторінки, також можна переглянути їх зміст.

Якщо це зображення, то буде відображене воно саме, якщо HTML-сторінка, можна буде переглянути її код, якщо файли CSS або JavaScript - то можна буде не тільки переглядати їх вміст, але й редагувати його, або, за необхідності, скопіювати код чи зберегти його як новий файл.

Також вкладка дозволяє створювати точки зупинки (Breakpoints) для налагодження коду.

6. Network

Ця вкладка дозволяє відстежити завантаження сторінки і всіх файлів, які ця сторінка підтягує при завантаженні (назва файлів, статус (відповідь сервера), тип файлу, ініціатори файлу, розмір файлу, час, за який він був завантажений).

Timeline – послідовність завантаження всіх файлів на сторінці. В нижньому рядку відображається кількість запитів, що були надіслані серверу, загальна кількість переданих даних, загальний час завантаження всіх файлів івсіх запитів.

DOMContentLoaded – час, за який буде завантажено весь HTML-код і побудоване DOM-дерево сторінок, Load – завантаження усіх необхідних ресурсів, що впливають на відображення сторінки.

6. Network

Секція «Request/response»:

- Request sent - час, за який запит було відправлено на сервер.
- Waiting TTFB (Time To First Byte) - час, за який сервер опрацював запит і відправив перший байт клієнту.
- Content Download - час завантаження контенту.

Секція «Connection Start»:

- Stalled - запит може бути зупинено з будь-якої причини, що описані в Queuing.

Секція «Resource scheduling»:

- Черга запитів в браузері, яка може утворитися через наступні причини:

● Існують запити з більш високим пріоритетом.

● Для цього джерела вже відкрито шість TCP-з'єднань, що є межею. Використовується тільки для HTTP / 1.0 и HTTP / 1.1.

● Браузер короткочасно розподіляє час для кешу.

6. Network

Якщо клікнути на файл із списку у вкладці Name, то відкриється додаткове вікно, в якому відображені властивості обраного елемента.

Коди відповіді сервера поділяються на 5 основних класів:

1xx – інформаційні коди, що відповідають за процес передачі даних. Вони тимчасові, інформують про те, що запит прийнято і обробка триває.

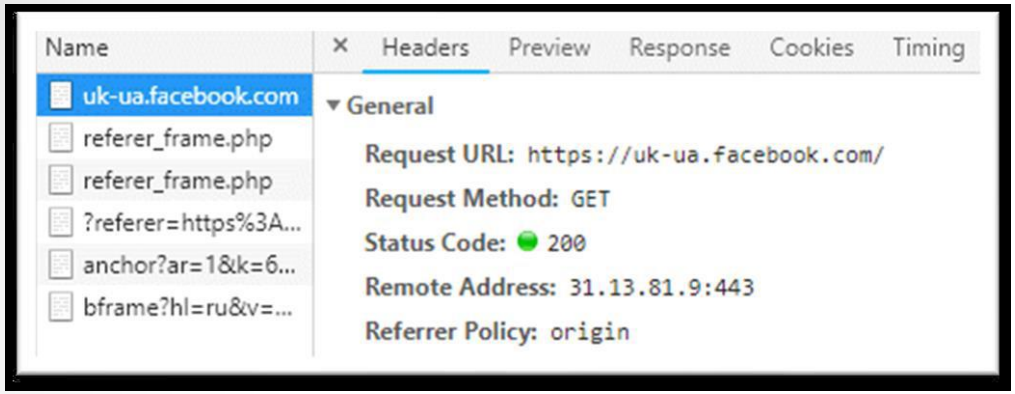
2xx – успішна обробка. Запит було отримано і успішно оброблено сервером.

3xx – перенаправлення (редирект). Ці відповіді сервера повідомляють про те, що необхідно виконати подальші дії для виконання запиту.

4xx – помилка користувача. Це означає, що запит не може бути виконаний з його вини.

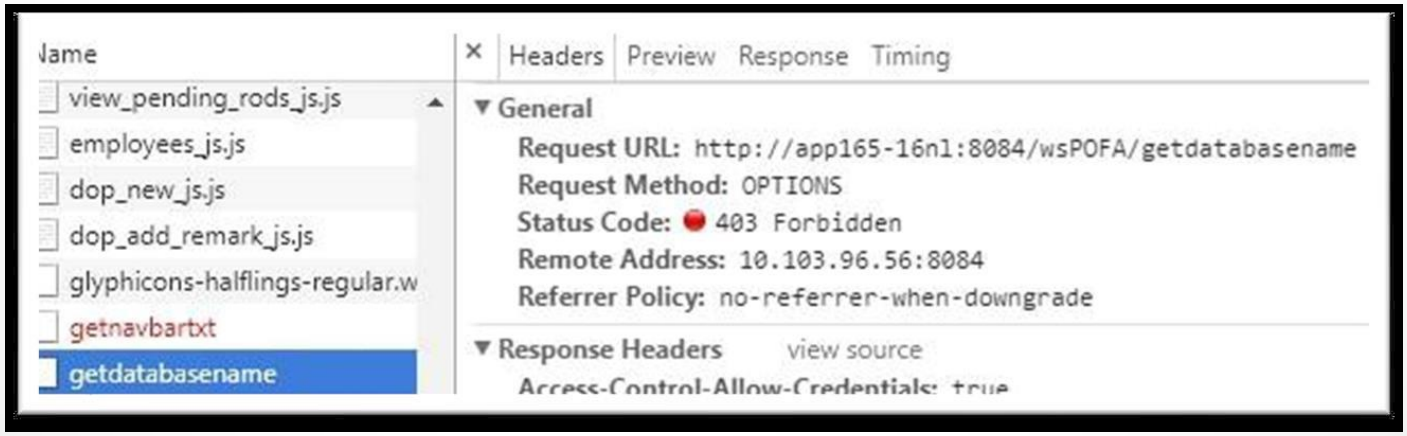
5xx – помилка сервера. Обов'язково відображається повідомлення, що не може опрацювати запит і з якої причини.

6. Network

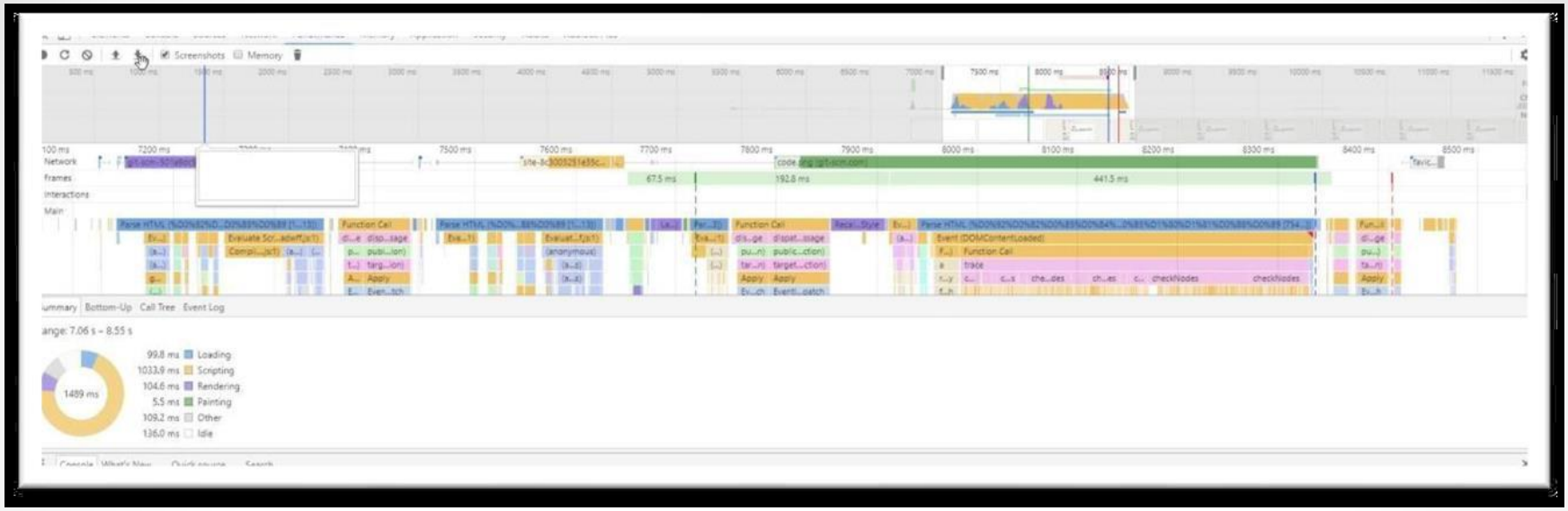


Наприклад, код зі статусом 200 ОК означає, що запит виконано успішно відповідно до очікувань користувача – потрібні дані існують і доступні для перегляду.

Код 403 Forbidden (Відмовлено у доступі) повертається у тому випадку, коли користувачу заборонений доступ до документа.

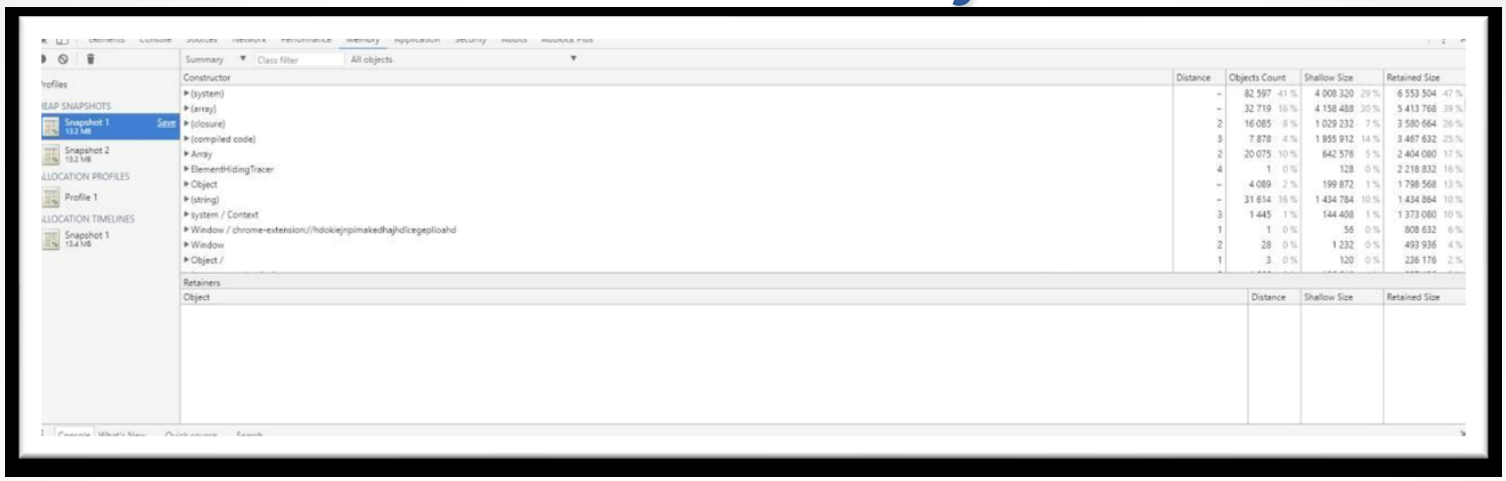


7. Performance



В цій вкладці також дублюється рядок Timeline, як і у вкладці Network, але в більш розгорнутому варіанті. Окрім мережі в цій вкладці можна відстежувати виконання JavaScript-коду і завантаження пам’яті, скільки процесорного часу це займає. Може використовуватися тестувальниками для перевірки продуктивності, швидкості завантаження сайту, пошуку слабких місць і можливих шляхів оптимізації.

8. Memory



Вкладка, за допомогою якої можна відслідковувати навантаження виконання JavaScript коду на пристрій, що тестується. Має три профілі:

- Heap snapshot – відображає розподіл пам’яті серед об’єктів в коді сторінки, а також пов’язані з ними DOM-елементи.
- Allocation instrumentation on timeline – відображає розподіл пам’яті серед змінних в коді сторінки, цей метод застосовується для ізоляції витоків пам’яті.
- Allocation sampling – відображає розподіл пам’яті на виконання JavaScript-функцій.

9. Application

Вкладка, в якій відображаються доступні сховища:

● Local storage – відображає, де можна переглянути, які змінні зберігає сторінка та їх значення, які можна змінити у разі необхідності.

● Session Storage – зберігаються дані поточної сесії.

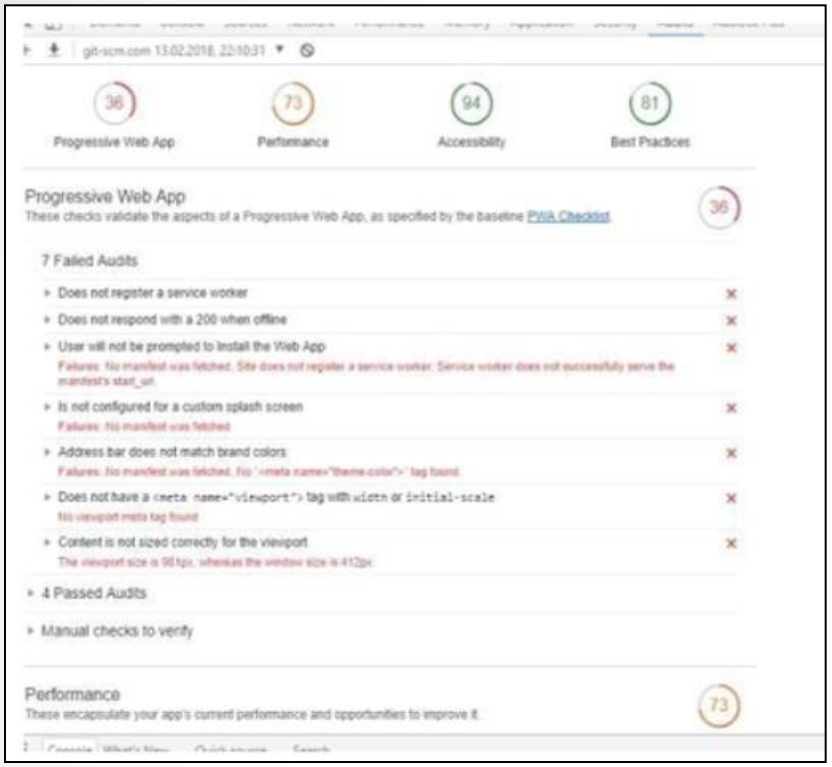
● Cookies – зберігаються всі дані cookies сайту.

Очистити сховища сайту можна за допомогою кнопки Clear. Очищення необхідно виконувати після оновлень версій продукту, оновлення файлів конфігурації, стилів, додавання/зміни функціональностей вебсайту та інших змін на сервері.

10. Security

Відображає протокол безпеки, якщо він існує, а також дані про сертифікат даного підключення.

11. Audits



Вкладка, яка допомагає визначити деякі проблеми сайту, що тестується.

Відбуваються перевірки того, чи ввімкнене стиснення, кешування, визначається, чи є невикористовувані CSS правила.

За допомогою цього інструмента можна провести аудит сайту і переглянути проблемні місця, що впливають на швидкість роботи. Можна провести аудит як для десктопної, так і для мобільної версії сайту, обравши відповідні параметри в налаштуваннях перед запуском.