**Що має знати Junior QA**

Web:

* Клієнт-серверна архітектура — базове розуміння;
* \*

Клієнт-серверна архітектура - це розподілена модель, яка використовується для побудови комп'ютерних систем та мереж. У цій архітектурі функції системи розділені між двома основними типами комп'ютерів: клієнтами і серверами. Розглянемо базове розуміння клієнт-серверної архітектури:

### Основні Компоненти:

1. **Клієнт:**
   * **Опис:**
     + Клієнт - це пристрій або програма, яка відправляє запити на сервер та отримує відповіді.
   * **Характеристики:**
     + Зазвичай має графічний інтерфейс користувача (GUI).
     + Виконується на кінцевому пристрої користувача, такому як комп'ютер, смартфон або планшет.
     + Ініціює запити до сервера.
2. **Сервер:**
   * **Опис:**
     + Сервер - це комп'ютер або програма, яка надає послуги або ресурси для клієнтів.
   * **Характеристики:**
     + Зазвичай розташований в центральному місці мережі.
     + Надає послуги, відповіді на запити клієнтів.

### Основні Принципи:

1. **Розділення Обов'язків:**
   * Клієнт і сервер виконують різні функції, що дозволяє розділити обов'язки та оптимізувати роботу системи.
2. **Комунікація:**
   * Клієнт і сервер обмінюються інформацією через мережу. Клієнт надсилає запити, а сервер надсилає відповіді.
3. **Спільний Інтерфейс:**
   * Клієнтські та серверні компоненти можуть використовувати стандартизований інтерфейс для взаємодії, що спрощує розробку та підтримку системи.

### Види Клієнт-Серверної Архітектури:

1. **Товстий Клієнт (Fat Client):**
   * Клієнт має значну частину логіки та функціональності.
   * Часто використовується в десктоп-додатках.
   * Вимагає мінімальної взаємодії з сервером.
2. **Тонкий Клієнт (Thin Client):**
   * Клієнт виконує мінімальну кількість обчислень.
   * Більша частина логіки зосереджена на сервері.
   * Часто використовується в веб-додатках.
3. **Централізована Архітектура:**
   * Усі обчислення та обробка даних відбуваються на сервері.
   * Клієнти використовуються лише для відображення та введення даних.
4. **Розподілена Архітектура:**
   * Логіка та обробка розподілені між клієнтом і сервером.
   * Клієнти виконують частину обчислень, але також здатні взаємодіяти з іншими клієнтами.

### Переваги та Недоліки:

#### Переваги:

* **Складність Системи:**
  + Дозволяє поділити роботу між клієнтом та сервером, що полегшує розробку та підтримку систем.
* **Масштабованість:**
  + Системи можуть легко масштабуватися, додаючи або модифікуючи клієнти та сервери.
* **Централізоване Управління:**
  + Дозволяє централізоване керування та підтримку серверів.

#### Недоліки:

* **Залежність від Мережі:**
  + Робота системи залежить від доступності та надійності мережі.
* **Потреба в Серверах Високої Міцності:**
  + Централізовані сервери можуть стати однією точкою відмови, особливо в потужних системах.
* **Безпека:**
  + Залежить від правильного налаштування безпеки на рівні мережі та серверів.

Клієнт-серверна архітектура є широко використовуваною моделлю для побудови розподілених систем та додатків у світі мережевого програмування.

* HTML, DOM-дерево, JS (на рівні розуміння, як будуються Web-сторінки).
* \*

HTML (Hypertext Markup Language), DOM (Document Object Model), та JavaScript (JS) є основними технологіями для побудови та взаємодії з веб-сторінками. Давайте розглянемо кожен з цих елементів та їх взаємодію:

**HTML (Hypertext Markup Language):**

1. **Опис:**
   * HTML є стандартною мовою розмітки для створення веб-сторінок.
   * Використовує теги для визначення структури та вмісту сторінки.
2. **Приклад Коду:**

html

1. <!DOCTYPE html>
2. <html>
3. <head>
4. <title>Заголовок сторінки</title>
5. </head>
6. <body>
7. <h1>Привіт, світ!</h1>
8. <p>Це моя перша веб-сторінка.</p>
9. </body>
10. </html>

**DOM (Document Object Model):**

1. **Опис:**
   * DOM - це програмний інтерфейс для представлення структури документа (веб-сторінки) у вигляді дерева об'єктів.
   * Дозволяє скриптовим мовам, таким як JavaScript, змінювати структуру, стиль та вміст веб-сторінки.
2. **Дерево Об'єктів:**
   * DOM представляє структуру сторінки як дерево об'єктів, де кожен тег та його вміст - це вузол.
3. **Взаємодія З JavaScript:**
   * Розробники можуть використовувати JavaScript для доступу до елементів DOM, зміни їх властивостей та створення нових елементів.

**JavaScript (JS):**

1. **Опис:**
   * JavaScript - це скриптова мова програмування, яка використовується для додавання інтерактивності до веб-сторінок.
2. **Взаємодія З HTML та DOM:**
   * JavaScript може бути вбудований безпосередньо в HTML-сторінку та взаємодіяти з DOM для зміни вмісту та структури сторінки.
3. **Приклад Коду:**

html

1. <!DOCTYPE html>
2. <html>
3. <body>
4. <h2>JavaScript у веб-сторінці</h2>
5. <p id="demo">JavaScript може змінювати вміст HTML-елементів.</p>
6. <button type="button" onclick="myFunction()">Натисни мене</button>
7. <script>
8. function myFunction() {
9. document.getElementById("demo").innerHTML = "Вітаю, JavaScript відпрацював!";
10. }
11. </script>
12. </body>
13. </html>

**Процес Побудови Веб-Сторінки:**

1. **Отримання HTML:**
   * Браузер отримує HTML-код від веб-сервера.
2. **Створення DOM-дерева:**
   * Браузер будує DOM-дерево, яке представляє структуру сторінки.
3. **Виконання JavaScript:**
   * Браузер виконує JavaScript-код, який може додати динамічний вміст, змінювати стилі та взаємодіяти з користувачем.
4. **Відображення Сторінки:**
   * Браузер відображає сторінку, використовуючи інформацію з DOM та CSS (Cascading Style Sheets).

В цілому, HTML, DOM та JavaScript спільно взаємодіють для створення інтерактивних та динамічних веб-сторінок, які можуть адаптуватися до дій користувача та змінюватися відповідно до різних сценаріїв.

Браузер:

* DEV tools (Network, console);
* \*

Інструменти розробника (Developer Tools) - це набір утиліт, які вбудовані в браузери та надають розробникам можливість аналізувати, відлагоджувати та оптимізовувати веб-сайти та веб-додатки. Декілька ключових вкладок у інструментах розробника, такі як Network та Console, дозволяють розробникам отримувати важливу інформацію та взаємодіяти з веб-сайтами. Давайте розглянемо ці вкладки:

### 1. Вкладка "Network" (Мережа):

Ця вкладка дозволяє вам моніторити мережевий трафік вашого веб-сайту, аналізувати запити та відповіді, вимірювати час завантаження ресурсів та виявляти проблеми зі швидкістю завантаження сторінки.

#### Функції та Можливості:

* **Запити та Відповіді:**
  + Відображає список всіх мережевих запитів та відповідей на сторінці.
  + Можливість фільтрації за типом ресурсу (HTML, CSS, JavaScript, зображення тощо).
* **Деталі Запитів:**
  + Детальна інформація про кожен запит, включаючи код відповіді, розмір, час виконання та інше.
* **Завантаження Ресурсів:**
  + Графік завантаження ресурсів дозволяє вам візуально аналізувати, як ресурси завантажуються та взаємодіють між собою.

### 2. Вкладка "Console" (Консоль):

Ця вкладка використовується для виведення повідомлень, помилок та виконання JavaScript-коду. Тут ви можете тестувати та відлагоджувати свій JavaScript-код.

#### Функції та Можливості:

* **Вивід Повідомлень:**
  + Повідомлення, які виводяться з JavaScript за допомогою console.log() або інших методів, відображаються тут.
* **Виконання Коду:**
  + Ви можете безпосередньо виконувати JavaScript-код у консолі.
* **Відлагодження та Помилки:**
  + Якщо є помилки у вашому коді, вони будуть відображені разом із стеком викликів та іншою інформацією.

### Як відкрити Інструменти Розробника:

1. **Google Chrome:**
   * Використовуйте клавішу F12 або правий клік на сторінці та оберіть "Інспектувати" або "Інструменти розробника".
2. **Mozilla Firefox:**
   * Використовуйте клавішу F12 або правий клік та оберіть "Інспектор елементів".
3. **Microsoft Edge:**
   * Використовуйте клавішу F12 або правий клік та оберіть "Інспектувати елемент".
4. **Safari:**
   * Перейдіть до "Увімкнути відлагодження" у налаштуваннях Safari та виберіть вкладку "Розробник" у меню.

Інструменти розробника є потужним засобом для аналізу та оптимізації роботи вашого веб-сайту або веб-додатка. Вони дозволяють розробникам ефективно відлагоджувати код, виявляти проблеми зі швидкістю та аналізувати мережевий трафік.

* Cookies.
* \*

Cookies (кукі, або HTTP cookies) - це невеликі фрагменти даних, які зберігаються на комп'ютері користувача та використовуються для збереження інформації про його активність на веб-сайті. Cookies використовуються для забезпечення персоналізації та покращення взаємодії з веб-сайтами. Ось основні аспекти, пов'язані з кукі:

**Основні Властивості Кукі:**

1. **Зберігання Даних:**
   * Cookies містять пари "ключ-значення" та можуть зберігати різноманітні інформаційні елементи, такі як ідентифікатори сесій, мови, налаштування користувача тощо.
2. **Термін Дії:**
   * Кожен cookie має термін дії, після закінчення якого він автоматично видаляється. Термін дії може бути встановлений на певний час або залежати від зміни налаштувань браузера.
3. **Безпека та Приватність:**
   * Cookies обмежені обсягом даних та можуть бути налаштовані для роботи тільки з певними доменами. Також, їх використання може бути обмежене політикою "SameSite" для покращення безпеки.
4. **Зберігання на Клієнтському Боці:**
   * Cookies зберігаються на комп'ютері або пристрої користувача та передаються на сервер з кожним запитом до відповідного домену.

**Використання Кукі:**

1. **Аутентифікація та Сесії:**
   * Кукі використовуються для збереження інформації про авторизацію та управління сесіями користувачів.
2. **Запам'ятовування Налаштувань:**
   * Використовуються для зберігання налаштувань користувача, таких як мова, тема тощо.
3. **Відстеження Активності:**
   * Cookies дозволяють веб-сайтам відстежувати активність користувачів, наприклад, аналіз відвідувань та взаємодії зі сторінками.
4. **Рекламні Цілі:**
   * Використовуються для відстеження користувачів з метою показу реклами, налаштованої на їхні інтереси.

**Запити та Відповіді Cookies:**

1. **Запит:**
   * Браузер передає cookies на сервер з кожним HTTP-запитом до відповідного домену.
2. **Відповідь:**
   * Сервер може відправляти нові або оновлені cookies разом з HTTP-відповіддю, які будуть зберігатися на клієнтському боці.

**Керування Cookies:**

1. **Створення Cookies:**
   * Cookies можуть бути створені сервером, встановлені на сторінці через JavaScript або збережені користувачем через браузер.
2. **Зміна та Видалення:**
   * Cookies можуть бути змінені або видалені з боку сервера під час відповіді на запит, або користувачем через налаштування браузера.
3. **Безпека та SameSite:**
   * SameSite - це атрибут, який може бути встановлений для cookies для обмеження їхнього використання на сторінках, які не є початковим доменом.

Куки - це важливий механізм для роботи зі станом сесії та покращення користувацького досвіду на веб-сайтах, але важливо дотримуватися рекомендацій з безпеки та приватності при їх використанні.

Тестування API:

* Rest API REST, SOAP API-архітектура (що це таке, різницю між ними, коди відповідей, методи);

\*

API (Application Programming Interface) тестування включає в себе перевірку функцій, надійності та продуктивності програмного інтерфейсу. У вашому питанні ви згадали дві популярні архітектури API - REST та SOAP. Давайте розглянемо їх основні характеристики:

**REST (Representational State Transfer):**

1. **Архітектура:**
   * **Основний принцип:** REST використовує принцип представлення ресурсу (Resource Representation). Кожен ресурс (наприклад, об'єкт бази даних або послуга) ідентифікується унікальним URI (Uniform Resource Identifier). Взаємодія з ресурсами відбувається через стандартні HTTP-методи (GET, POST, PUT, DELETE).
   * **Представлення:** Дані можуть бути представлені у форматі JSON або XML.
   * **Стан:** REST є станово-орієнтованим, що означає, що кожен запит від клієнта повинен містити всю інформацію, необхідну для розуміння та обробки запиту.
2. **Методи HTTP:**
   * GET: Отримати дані з ресурсу.
   * POST: Створити новий ресурс або здійснити дію.
   * PUT: Оновити існуючий ресурс або створити, якщо він не існує.
   * DELETE: Видалити ресурс.
3. **Коди відповіді HTTP:**
   * 200 OK: Успішний запит.
   * 201 Created: Ресурс створений успішно.
   * 204 No Content: Успішний запит без тіла відповіді.
   * 400 Bad Request: Некоректний або невірний запит.
   * 401 Unauthorized: Потрібна аутентифікація для доступу.
   * 404 Not Found: Ресурс не знайдено.
   * 500 Internal Server Error: Помилка на сервері.

**SOAP (Simple Object Access Protocol):**

1. **Архітектура:**
   * **Основний принцип:** SOAP базується на обміні повідомленнями у форматі XML через мережу, часто використовуючи протокол HTTP або SMTP.
   * **Представлення:** Дані завжди представлені у форматі XML.
   * **Стан:** SOAP може бути як станово-орієнтованим, так і безстаним.
2. **Методи HTTP:**
   * Методи HTTP не обов'язкові, оскільки SOAP може використовувати різні транспортні протоколи.
3. **Коди відповіді HTTP:**
   * 200 OK: Успішний запит.
   * 201 Created: Ресурс створений успішно.
   * 500 Internal Server Error: Помилка на сервері.

**Вибір між REST і SOAP:**

* REST є легшим для розуміння та використання, особливо для веб-розробників.
* SOAP надає більше можливостей для реалізації складних бізнес-процесів та має більше вбудованих механізмів безпеки.

**Тестування API:**

1. **Функціональне тестування:**
   * Перевірка правильності виконання HTTP-методів.
   * Перевірка кодів відповіді HTTP.
   * Перевірка формату та структури даних у відповідях (JSON або XML).
2. **Надійність:**
   * Тестування обробки помилок та некоректних запитів.
   * Тестування обробки та відновлення стану після відмови.
3. **Продуктивність:**
   * Тестування швидкодії викликів API.
   * Тестування великих обсягів даних та високих навантажень.
4. **Безпека:**
   * Тестування автентифікації та авторизації.
   * Тестування захисту від атак, таких як SQL ін'єкції та перехоплення сесій.
5. **Тестування меж ресурсів (Cross-resource testing):**
   * Перевірка взаємодії між різними ресурсами та методами.

* JSON, XML (що таке, навіщо потрібно).
* \*

SON (JavaScript Object Notation) та XML (eXtensible Markup Language) - це два різних формати обміну даними, які використовуються для передачі структурованих інформаційних даних між комп'ютерами. Обидва формати мають свої особливості та застосування.

**JSON (JavaScript Object Notation):**

1. **Що таке JSON:**
   * JSON - це легкий текстовий формат обміну даними, який використовується для представлення структурованих даних у вигляді пар "ключ-значення". В основі JSON лежить синтаксис мови програмування JavaScript, але він став популярним і в інших мовах.
2. **Формат:**
   * Виглядає як набір пар "ключ-значення" в фігурних дужках.

json

{

"name": "John Doe",

"age": 30,

"city": "New York"

}

**3. Навіщо потрібен JSON:**

* + **Легкість читання і запису:** JSON легко читається і записується як людьми, так і машинами.
  + **Мовно-нейтральний:** Може бути використаний в різних мовах програмування.
  + **Підтримка об'єктів та масивів:** Дозволяє структурувати дані у вигляді об'єктів та масивів.

**XML (eXtensible Markup Language):**

1. **Що таке XML:**
   * XML - це розширюваний мовний формат, призначений для представлення структурованих даних у вигляді тексту. XML використовує теги для визначення структури та семантики даних.
2. **Формат:**
   * Виглядає як дерево тегів.

xml

<person>

<name>John Doe</name>

<age>30</age>

<city>New York</city>

</person>

**3. Навіщо потрібен XML:**

* + **Універсальність:** Може бути використаний для будь-якого типу даних та мов програмування.
  + **Розширюваність:** Дозволяє визначати власні теги та розширювати мову для конкретних вимог.
  + **Підтримка структури:** Дані можуть бути структуровані за допомогою вкладених тегів.

**Вибір між JSON та XML:**

* **JSON:**
  + Легший для читання та написання.
  + Ідеальний для використання в мовах програмування JavaScript.
  + Поширено в веб-розробці та API.
* **XML:**
  + Дозволяє більшу гнучкість та розширюваність.
  + Ідеальний для документів зі складною структурою.
  + Використовується в різних областях, включаючи документообіг та конфігураційні файли.

Інструменти:

* Postman, Fiddler;
* JMeter (для перформанс-тестування).

Робота з базами даних:

* SQL (на рівні базових SELECT з різними умовами).

Інструменти:

* SQL Server Management Studio (або альтернативи).

Трекери:

* Jira (або альтернативи) — для створення задач, тест-кейсів, багів;
* TestRail (або альтернативи).

DNS (Domain Name System) — базове розуміння:

* Доменні імена: розуміти, як доменні імена призначаються та інтерпретуються. Доменні імена використовують для ідентифікації ресурсів в інтернеті (наприклад, [www.example.com](http://www.example.com/)).
* \*

**Доменне ім'я** (Domain Name) - це читабельне та легко запам'ятовуване ім'я, яке використовується для ідентифікації ресурсів в Інтернеті, таких як веб-сайти, електронна пошта, сервери тощо. Доменні імена використовуються для заміни числових IP-адрес і полегшення взаємодії користувачів з різними ресурсами.

**Структура доменного імені:**

Доменні імена мають ієрархічну структуру та складаються з послідовності піддоменів, розділених крапками. Найвищий рівень домену (Top-Level Domain, TLD) розташований в кінці імені.

Наприклад, у доменному імені www.example.com:

* www - піддомен (може бути відсутнім).
* example - другорядний домен (Second-Level Domain, SLD).
* com - TLD.

**Процес призначення доменного імені:**

1. **Вибір та Реєстрація:**
   * Користувач вибирає доменне ім'я та реєструє його через реєстратора доменів або через іншого провайдера доменів.
2. **Визначення Типу Домену:**
   * Вибір типу домену (TLD), наприклад, .com, .org, .net, .gov, .edu, .uk, тощо.
3. **Введення Данних:**
   * Введення особистих або організаційних даних відповідно до вимог реєстратора.
4. **Оплата та Реєстрація:**
   * Оплата реєстраційного збору за обране доменне ім'я.
5. **Підтвердження та Активація:**
   * Підтвердження реєстрації та активація домену.

**Інтерпретація Доменного Імені:**

1. **Розрізнення на Комп'ютері:**
   * Коли ви вводите домен у веб-браузер, DNS (Domain Name System) перетворює доменне ім'я на відповідну IP-адресу, яка вказує на сервер, де розташований ресурс.
2. **Ієрархія DNS:**
   * DNS використовує ієрархічну структуру для знаходження відповідності доменних імен і IP-адрес. Запити шлються від кореневого DNS-сервера до конкретного сервера, який відповідає за певний TLD та вниз по ієрархії до сервера, який відповідає за конкретний домен.
3. **Управління DNS-записами:**
   * Власник домену може налаштовувати різноманітні DNS-записи, такі як A-записи (для вказання IP-адреси), MX-записи (для налаштування поштового обміну), CNAME-записи (для створення альтернативних імен), тощо.

Доменні імена грають важливу роль в Інтернеті, забезпечуючи зручний та підтримуваний спосіб доступу до веб-ресурсів.

* DNS-запити: як браузери перетворюють доменні імена на IP-адреси за допомогою DNS-запитів.
* \*

DNS (Domain Name System) є ключовою складовою Інтернету, яка використовується для перетворення людських-читабельних доменних імен на IP-адреси, які комп'ютери в мережі можуть розуміти. Коли ви вводите доменне ім'я у веб-браузері, відбувається процес DNS-запиту. Давайте розглянемо, як це відбувається:

### 1. Запитання від Користувача:

1. **Введення Домену:**
   * Користувач вводить доменне ім'я у веб-браузер (наприклад, [www.example.com](http://www.example.com)) та натискає "Enter".

### 2. Пошук в Локальному Кеші:

1. **Перевірка Локального Кешу:**
   * Браузер перевіряє свій локальний DNS-кеш, щоб визначити, чи є відповідь на це доменне ім'я вже збережено в кеші.

### 3. Запит до DNS-Сервера:

1. **Запит до Резольвера:**
   * Якщо доменне ім'я не знайдено в локальному кеші, браузер направляє DNS-запит до DNS-резольвера, який зазвичай надається вашим Інтернет-постачальником.

### 4. Пошук у Ієрархії DNS:

1. **Пошук у Кореневому DNS-Сервері:**
   * Резольвер спочатку направляє запит до кореневого DNS-сервера, який надає вказівки для пошуку відповідного TLD-сервера.
2. **Запит до TLD-Сервера:**
   * Кореневий DNS-сервер вказує на TLD-сервер для конкретного домену (наприклад, .com, .org, тощо). Резольвер направляє запит до TLD-сервера.

### 5. Запит до Авторитетного DNS-Сервера:

1. **Запит до Авторитетного DNS-Сервера:**
   * TLD-сервер вказує на авторитетний DNS-сервер для конкретного домену (наприклад, для example.com). Резольвер направляє запит до цього авторитетного DNS-сервера.

### 6. Отримання IP-Адреси:

1. **Отримання IP-Адреси:**
   * Авторитетний DNS-сервер повертає IP-адресу, пов'язану з вказаним доменним іменем, резольверу.

### 7. Повернення Результату Користувачеві:

1. **Відправлення IP-Адреси Браузеру:**
   * Отримавши IP-адресу від авторитетного DNS-сервера, резольвер повертає цю інформацію в браузер.
2. **Запит до Веб-Сервера:**
   * Браузер використовує отриману IP-адресу для створення запиту до веб-сервера, який хостить вміст для вказаного домену.

### 8. Відображення Веб-Сторінки:

1. **Отримання Веб-Сторінки:**
   * Веб-сервер відповідає на запит вмістом веб-сторінки, і цей вміст відображається в браузері користувача.

Цей процес називається розподіленим пошуком та резольвінгом DNS, і він забезпечує ефективний механізм перетворення доменних імен на IP-адреси в Інтернеті.

* Протоколи передачі даних — базове розуміння:
* HTTP і HTTPS: розуміння протоколів для передачі гіпертекстового змісту через інтернет.
* \*

HTTP (Hypertext Transfer Protocol) і HTTPS (Hypertext Transfer Protocol Secure) є протоколами для передачі гіпертекстового змісту через Інтернет. Обидва протоколи використовуються для обміну даними між веб-клієнтом (наприклад, веб-браузером) та веб-сервером. Давайте розглянемо основні відмінності між HTTP і HTTPS.

**HTTP (Hypertext Transfer Protocol):**

1. **Незахищений протокол:**
   * HTTP передає дані у відкритому текстовому форматі, що робить їх вразливими до перехоплення та зчитування з боку зловмисників.
2. **Використання порту 80:**
   * HTTP використовує порт 80 для обміну даними між клієнтом і сервером.
3. **Брак шифрування:**
   * HTTP не надає шифрування для передачі даних. Інформація, така як облікові записи та паролі, може бути перехоплена іншими користувачами.
4. **Необхідність удосконалень для безпеки:**
   * Для забезпечення безпеки використовуються додаткові заходи, такі як HTTPS або використання VPN (Virtual Private Network).

**HTTPS (Hypertext Transfer Protocol Secure):**

1. **Захищений протокол:**
   * HTTPS використовує шифрування (зазвичай TLS або SSL) для захисту передаваних даних від несанкціонованого доступу та перехоплення.
2. **Використання порту 443:**
   * HTTPS використовує порт 443 для забезпечення захищеної комунікації між клієнтом і сервером.
3. **Достовірність та аутентифікація:**
   * HTTPS включає процес аутентифікації веб-сайту за допомогою сертифікатів, що дозволяє користувачеві впевнитися в тому, що він спілкується з правильним сервером.
4. **Захист від MITM-атак:**
   * HTTPS ускладнює атаки "Man-in-the-Middle" (MITM), оскільки шифровані дані не можуть легко перехоплюватися або змінюватися зловмисниками.

**Важливі відмінності:**

* **Безпека:**
  + Основна відмінність між HTTP і HTTPS полягає в безпеці передаваних даних. HTTPS забезпечує шифрування, що робить комунікацію безпечною.
* **Шифрування:**
  + HTTP не шифрує дані, тоді як HTTPS використовує шифрування для захисту конфіденційної інформації.
* **Сертифікати:**
  + HTTPS вимагає встановлення SSL- або TLS-сертифікату для підтвердження ідентифікації сервера.
* **Використання портів:**
  + HTTP використовує порт 80, в той час як HTTPS використовує порт 443.

**Вибір між HTTP і HTTPS:**

З огляду на важливість безпеки в Інтернеті, використання HTTPS стає все більш стандартним. Багато браузерів тепер позначають веб-сайти без HTTPS як "небезпечні". Тому в реальних веб-проектах рекомендується використовувати HTTPS для захисту конфіденційності і цілісності даних, переданих між клієнтом і сервером.

* TCP/IP (Transmission Control Protocol/Internet Protocol): основний стек протоколів для передачі даних в інтернеті.
* \*

CP/IP (Transmission Control Protocol/Internet Protocol) представляє собою основний стек протоколів для передачі даних в мережі Інтернет. Цей стек протоколів включає в себе набір стандартів, які дозволяють різним комп'ютерам та мережевим пристроям ефективно обмінюватися інформацією. Давайте розглянемо ключові протоколи, які утворюють стек TCP/IP:

**1. IP (Internet Protocol):**

* **Опис:**
  + IP визначає правила для адресації та маршрутизації пакетів даних між комп'ютерами в мережі.
* **Версії:**
  + IP може бути в двох версіях - IPv4 (зазвичай використовується) та IPv6 (новіший та має більше доступних адрес).

**2. TCP (Transmission Control Protocol):**

* **Опис:**
  + TCP відповідає за передачу даних у надійному та послідовному порядку між пристроями у мережі.
  + Забезпечує контроль за з'єднанням, розбиттям та складанням повідомлень.
* **Особливості:**
  + Надійність, відновлення з'єднання, контроль потоку.

**3. UDP (User Datagram Protocol):**

* **Опис:**
  + UDP - це протокол передачі даних без встановлення з'єднання, менш вимогливий до ресурсів порівняно з TCP.
  + Використовується для передачі даних, коли точна доставка менш критична (наприклад, відео- або голосовий стрімінг).
* **Особливості:**
  + Без встановлення з'єднання, відсутність гарантії доставки.

**4. ICMP (Internet Control Message Protocol):**

* **Опис:**
  + ICMP використовується для відправки повідомлень про помилки та для діагностики мережевих проблем.
  + Часто використовується інструментами, такими як ping для тестування доступності вузлів мережі.

**5. IPSec (Internet Protocol Security):**

* **Опис:**
  + IPSec надає безпеку для IP-з'єднань шляхом шифрування та аутентифікації даних, які пересилаються через мережу.

**6. ARP (Address Resolution Protocol):**

* **Опис:**
  + ARP використовується для знаходження асоційованої з IP-адресою мережевої карти MAC-адреси в локальній мережі.

**7. DHCP (Dynamic Host Configuration Protocol):**

* **Опис:**
  + DHCP дозволяє автоматично призначати IP-адреси та інші мережеві конфігурації пристроям у мережі.

**8. DNS (Domain Name System):**

* **Опис:**
  + DNS перетворює доменні імена в IP-адреси та забезпечує службу розподіленого об'єктно-орієнтованого зберігання інформації.

**Важливі аспекти TCP/IP:**

* **Основна Інфраструктура:**
  + TCP/IP є основою Інтернету та використовується у більшості мережевих середовищ.
* **Мережева Ідентифікація:**
  + IP-адреса є унікальним ідентифікатором пристрою у мережі.
* **Передача Даних:**
  + TCP та UDP використовуються для передачі даних між пристроями.
* **Безпека та Контроль:**
  + Протоколи, такі як IPSec, забезпечують безпеку та конфіденційність передачі даних.
* **Мережева Архітектура:**
  + TCP/IP використовується в різних мережевих архітектурах, таких як LAN, WAN, та Інтернет.

Стек протоколів TCP/IP є важливою основою для забезпечення ефективного та надійного обміну даними в Інтернеті.

**Де шукати знання новачку QA**

**YouTube-канали:**

* [QA Club Lviv](https://www.youtube.com/@QAClubLviv)
* [qa семпай | автоматизація тестування](https://www.youtube.com/@qa_senpai)
* [Попелюха Тестування ПЗ](https://www.youtube.com/@Popeliuha)

**Безплатні курси:**

* [REDROVER.SCHOOL](https://redrover.school/#submenu:more) (різні напрями для QA)
* [Курс з Git на Prometheus](https://prometheus.org.ua/course/course-v1:LinuxFoundation+GIT101+2023_T1)
* [Test Automation University](https://testautomationu.applitools.com/) (інструменти та фреймворки для автоматизаторів)

**Платформи**

Для ознайомлення з основною термінологією та принципами тестування: [Guru99. Software Testing Tutorial](https://www.guru99.com/software-testing.html)

Для практикування навичок написання коду: [LeetCode](https://leetcode.com/)

**Що почитати про професію QA на DOU**

* [Співбесіда з QA. 250+ запитань для Junior, Middle, Senior](https://dou.ua/lenta/articles/interview-qa/)
* [Зарплати українських тестувальників — літо 2023](https://dou.ua/lenta/articles/salary-report-qa-summer-2023/)
* [«У світчерів майже завжди є перевага». Поради тестувальникам-початківцям від Senior QA Engineers](https://dou.ua/lenta/articles/qa-advices/)
* [«Помітила, що пів року мені не писали рекрутери в LinkedIn». Як досвідчені QA-фахівці шукають роботу і оцінюють ситуацію на ринку вакансій](https://dou.ua/lenta/articles/middle-and-senior-qa-specialists-about-job-market/)
* [Ціна роботи. Скільки коштують робочі інструменти та освіта QA](https://dou.ua/lenta/articles/price-of-work-qa/)
* [Синергія розробника і тестувальника: як запобігти негативному впливу когнітивних упереджень](https://dou.ua/lenta/articles/tests-driven-by-developers/)
* [Помилка ціною в десятки тисяч і баг, що з’являється двічі на рік. Українські QA — про нестандартні проблеми, з якими стикалися в роботі](https://dou.ua/lenta/articles/qa-about-work-bugs/)