**IT theory**

### **background-repeat**

За замовчуванням, фонове зображення буде повторюватися нескінченно по вертикалі та горизонталі, якщо не вказано інше. Властивість background-repeat може бути використана, щоб змінити напрямок, в якому фонова картинка буде повторюватися, якщо вона взагалі повторюється.

div {  
 background-image: url("image.png");  
 background-repeat: no-repeat;  
}

Властивість background-repeat приймає чотири різних значення: repeat, repeat-x, repeat-y і no-repeat. repeat є значенням за замовчуванням і повторює фонове зображення по вертикалі та по горизонталі.

Значення repeat-x повторює фонове зображення по горизонталі, тоді як значення repeat-y повторює його по вертикалі. Нарешті, значення no-repeat говорить браузеру відобразити картинку один раз, тобто не повторювати її взагалі.

### **background-position**

За замовчуванням, фонова картинка розташовується в лівому верхньому кутку елемента. Проте за допомогою властивості background-position ми можемо контролювати, де саме зображення розміщується відносно цього кута.

div {  
 background-image: url("image.png");  
 background-position: 20px 10px;  
 background-repeat: no-repeat;  
}

Властивість background-position вимагає два значення: горизонтальне зміщення (перше значення) і вертикальне зміщення (друге значення). Якщо вказано тільки одне значення, то воно застосовується для горизонтального зміщення, а вертикальне за замовчуванням задається як 50% (center).

Оскільки ми рухаємо фонове зображення з лівого верхнього кута елемента, значення розміру будуть безпосередньо прив'язані до цього кута.

Щоб встановити значення background-position ми можемо використовувати ключові слова top, right, bottom і left, пікселі, відсотки або будь-яку одиницю розміру. Ключові слова і відсотки працюють дуже схоже. Значення left top ідентичне відсотковому значенню 0 0, яке розташовує зображення в лівому верхньому кутку елемента. Значення right bottom ідентичне відсотковому запису 100% 100% і позиціонуватиме зображення в правому нижньому кутку елемента.

## **background-size**

Властивість background-size дає змогу вказати розмір фонового зображення. Ця властивість приймає кілька значень, зокрема розмір і ключові слова.

У разі використання розміру ми можемо вказати ширину і висоту за допомогою двох значень розділених пропуском. Перше значення встановить ширину фонового зображення, а друге значення його висоту. Важливо зазначити, що відсотковий запис пов'язаний з розміром елемента, а не з вихідним розміром фонової картинки.

Отже, властивість background-size із шириною 100% встановить, що фонове зображення займає всю ширину елемента. Якщо друге значення не визначено після ширини, то значення висоти буде задано автоматично, щоб зберегти пропорції фонового зображення.

Ключове слово auto може бути задіяно як значення ширини або висоти, щоб зберегти пропорції фонового зображення. Так, якщо ми хочемо встановити висоту фонової картинки на 75% від висоти елемента при збереженні її пропорцій, то можемо задати значення background-size як auto 75%.

**Error**

div {  
 background: url("image.jpg") 0 0 no-repeat;  
 background-size: auto 75%;  
}**Error**

Ключові слова cover і contain На додаток до розмірів властивість background-size також допускає ключові слова cover і contain.

Значення cover вказує, що фонове зображення буде змінено, щоб повністю закрити ширину і висоту елемента. Початкове співвідношення сторін фонової картинки буде збережено, поки зображення розтягуватиметься або стискатиметься в міру необхідності, щоб охопити елемент цілком. Часто під час використання значення cover частина фонового зображення обрізається, щоб зображення займало всю доступну площу елемента.

Ключове слово contain, з іншого боку, вказує, що фонове зображення буде змінено, щоб цілком вписатися в ширину і висоту елемента. При цьому вихідне співвідношення сторін фонового зображення буде збережено, але картинка розтягуватиметься або стискатиметься в міру необхідності, щоб залишитися в межах ширини і висоти елемента. На відміну від cover, ключове слово contain завжди показує повне фонове зображення, водночас часто воно не буде займати всю доступну площу елемента.

Значення cover і contain можуть призвести до невеликих викривлень фонових зображень, особливо, коли картинка розтягується за межами своїх початкових розмірів. Ми повинні стежити за цим під час використання цих значень і переконатися, що стилі в результаті є відповідними.

### **width: auto**

* **Описание**: Значение auto для свойства width означает, что элемент займет всю доступную ширину в контейнере, за исключением значений, установленных свойствами margin, border и padding. Если родительский элемент имеет фиксированную ширину, то элемент с width: auto будет растягиваться на всю доступную ширину.

### **height: auto**

* **Описание**: Значение auto для свойства height означает, что высота элемента будет определяться его содержимым. Элемент будет растягиваться вертикально, чтобы вместить весь свой контент.

## **background-clip і background-origin**

Властивість background-clip визначає площу поверхні, яку охоплює фонове зображення, а властивість background-origin визначає, звідки має починатися background-position. Впровадження цих двох нових властивостей пов'язане з впровадженням трьох нових значень: border-box, padding-box і content-box. Кожне з цих трьох значень може бути використане для властивостей background-clip і background-origin.

**Error**

div {  
 background: url("shay.jpg") 0 0 no-repeat;  
 background-clip: padding-box;  
 background-origin: border-box;  
}**Error**

Значення властивості background-clip за замовчуванням встановлюється як border-box, що дає змогу фоновому зображенню розширюватися в область меж. Тим часом, властивість background-origin за замовчуванням встановлюється як padding-box, що дає змогу фоновому зображенню починатися там само, де і padding в елемента.

Ми вже обговорювали значення цих ключових слів, коли розглядали властивість box-sizing. Сенс самих значень не змінився, але їхні функції змінюються під час використання різних властивостей фону.

## **img або background?**

Обидва способи - використання тега **<img>** та фонів CSS - можуть використовуватися для відображення зображень на веб-сторінці, але вони мають свої особливості та використовуються в різних ситуаціях.

### **Тег <img>**

Тег **<img>** використовується для вставки зображень безпосередньо в HTML-код. Наприклад:

**Error**

<img src="image.jpg" alt="Опис зображення">**Error**

**Індексація пошуковими системами:Тег <img>:** Зображення, вставлене за допомогою тега **<img>**, індексується пошуковими системами. Альтернативний текст, вказаний у властивості **alt**, може бути важливим для SEO (оптимізації для пошукових систем), оскільки пошукові системи можуть використовувати цей текст для розуміння змісту зображення.

**Доступність:Тег <img>:** Забезпечує можливість встановлювати альтернативний текст для зображення, що полегшує сприйняття контенту користувачами з обмеженими можливостями.

**Взаємодія з JavaScript:Тег <img>:** Зображення може бути використане для взаємодії з JavaScript, наприклад, ви можете змінювати джерело зображення динамічно за допомогою скриптів.

**Завантаження зображення:Тег <img>:** Браузер завантажує зображення безпосередньо при завантаженні сторінки, що може впливати на швидкість завантаження сторінки.

**Error**

### **Фони CSS**

Фони CSS можна встановити для будь-якого елемента в HTML за допомогою властивостей CSS, таких як background-image. Наприклад:

**Error**

div {  
 width: 300px;  
 height: 200px;  
 background-image: url('image.jpg');  
 background-size: cover;  
}**Error**

**Фони CSS:** Зображення, використане як фон через CSS, може бути менш ефективним для індексації пошуковими системами, оскільки пошукові системи можуть ігнорувати фонові зображення. Тому, якщо важливо, щоб зображення було індексоване пошуковими системами, використання тега **<img>** може бути більш доцільним.

**Доступність:Фони CSS:** Фони не надають такого рівня доступності, як тег **<img>**. Немає спеціальної можливості додавати альтернативний текст безпосередньо для фонового зображення.

**Взаємодія з JavaScript:Фони CSS:** Взаємодія з фонами через JavaScript може бути менш простою, іншими словами, фони не можуть бути змінені так легко, як зображення, вставлені через тег **<img>**.

**Завантаження зображення:Фони CSS:** Зображення, використані як фон, завантажуються за потребою, що може поліпшити швидкість завантаження сторінки, особливо, якщо ви використовуєте оптимізовані формати зображень.

**Error**

### **Коли використовувати який метод?**

Використовуйте тег **<img>**, якщо вам потрібно:

Індексацію пошуковими системами для зображення.

Взаємодію з JavaScript, яка потребує зміни джерела зображення.

Високий рівень доступності для користувачів з обмеженими можливостями.

2. Використовуйте фони CSS, якщо:

Вам не потрібна індексація пошуковими системами для конкретного зображення.

Ви хочете додатковий контроль над параметрами фону, такими як розмір, повторення та позиція.

Швидкість завантаження є важливою, і ви хочете використовувати оптимізовані фонові зображення.

У кожному конкретному випадку важливо враховувати потреби вашого проекту та вибрати відповідний підхід.

## **SVG. Як його використовувати.Quiz. ДЗ**

\*\*SVG\*\* - це формат векторної графіки, дослівно: масштабована векторна графіка. У векторних форматах зберігається не саме зображення, а інструкція з його побудови по точках і кривих.

У растрових форматах інформація про конкретну кількість точок зображення щільно упакована в бінарну цеглу. У неї марно заглядати і змінювати її можна тільки в редакторах графіки.

Формат SVG теж можна створювати і змінювати в редакторах графіки, на кшталт Illustrator, Sketch або

Я вам більше скажу: SVG - це як окрема HTML-сторінка. Коли ви вставляєте SVG, ви, насправді, вставляєте не просто картинку, а цілу сторінку. Зі своєю системою координат, вьюпортом, стилями, скриптами і дивовижними особливостями.

Якщо дивитися на SVG як на окрему сторінку - стає зрозуміліше, який спосіб вставки вам потрібен. Є чотири основних і в кожного - особливості.

Перший і найпростіший - елемент img прямо в HTML-коді. Це в принципі найефективніший спосіб завантажити будь-яку картинку - браузери заздалегідь знають за HTML-кодом, що вона є, і починають її довантажувати.

Мінус у тому, що в такому SVG не працюватимуть скрипти і будь-які спроби взаємодії з елементами всередині приречені. Файл буде як за склом: дивитися можна, а чіпати не можна. Хоча всередині все інше прекрасно працює, включно з CSS-анімаціями.

акий спосіб найкраще підходить контентним зображенням, яким не потрібна взаємодія: логотипи, графіки, схеми.

Другий спосіб - фонова картинка в CSS. Причому неважливо, задасте ви його елементу, псевдоелементу або контентом вставите - результат буде таким самим, як з img: за склом, але всередині щось працює.

Цей спосіб підходить для оформлювальної графіки, якій не потрібна взаємодія: фони, іконки та інші дрібниці.

Третій спосіб, через object, нарешті вибиває скло між сторінкою і нутрощами SVG-файлу. Працюють скрипти, взаємодія, анімація - якщо вони описані всередині SVG. Між тегами object можна вставити фолбек, який з'явиться, якщо браузер не говорить на SVG.

Насправді, замість object можна навіть використовувати iframe, як якщо б ви підключили іншу сторінку. Але object працює краще і підлаштовується під розміри картинки.

За гнучкість доводиться платити: через те, що це вже не просто графіка і там можна скриптувати, до такого способу висуваються інші вимоги безпеки. Наприклад, картинку з іншого домену просто так уже не вставити.

Цей спосіб підходить, коли вам потрібно вставити якусь інтерактивну графіку: іграшки, графіки і всяке складне. Досить згадати, що колись через object вставлялися Flash-ролики. Запитайте в батьків, що це таке.

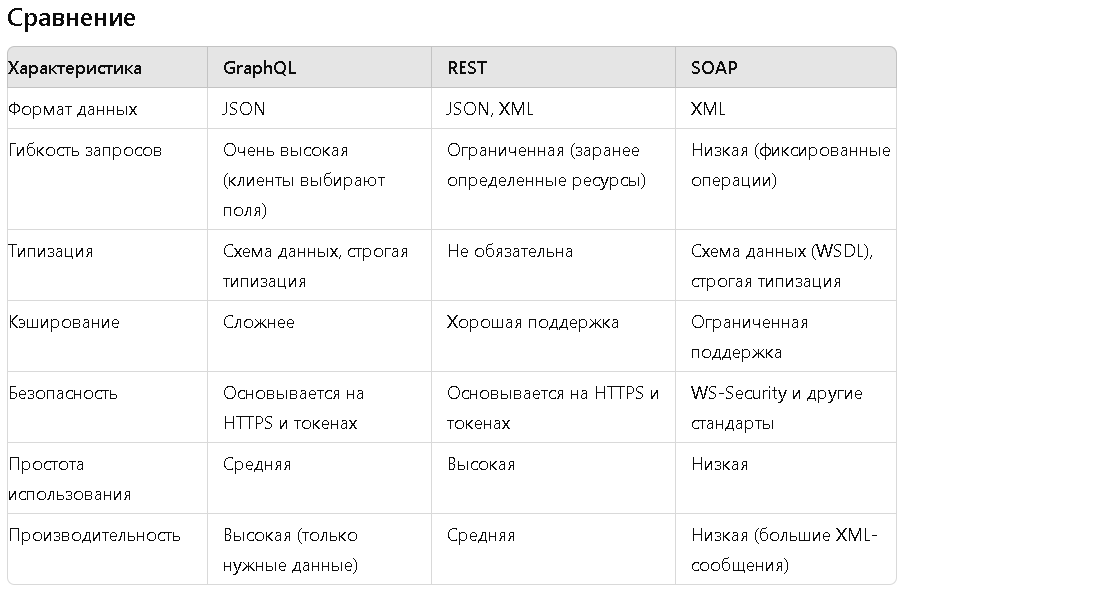
Четвертий спосіб запрацював, коли браузери переписали свої HTML-парсери за новим стандартом, і вміст SVG-файлів стало можна вставляти прямо на сторінку, як будь-які інші теги.

З таким SVG можна робити те саме, що і зі звичайними HTML-елементами: стилі, скрипти - ну, ви самі знаєте. Можна, наприклад, змінювати колір заливки під час наведення й описувати все в загальних стилях.

Мінус у тому, що такі картинки не кешуються окремо від сторінки - хоча це можна обійти через символи і юзи, але це довга історія, ми про це ще поговоримо окремо.

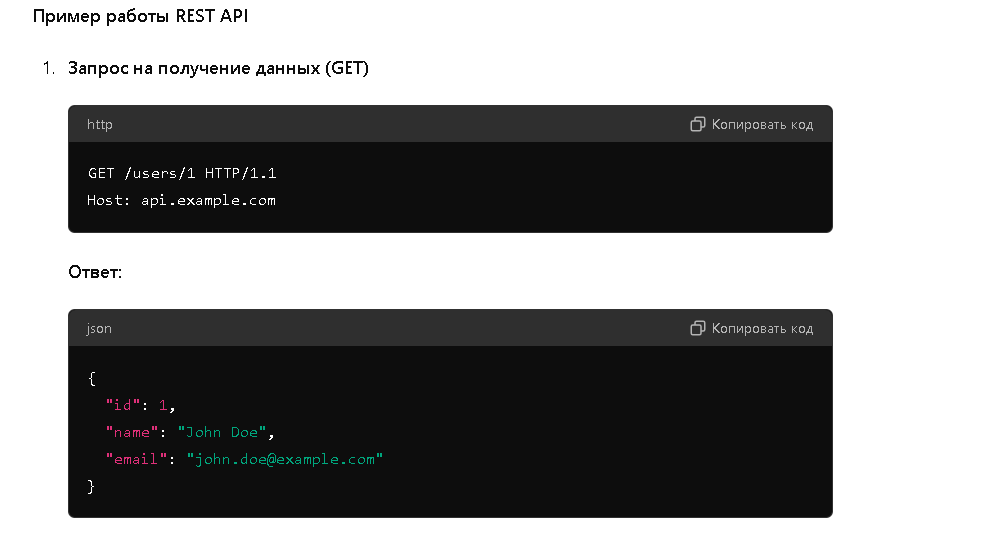
У підсумку: способів купа і всі чимось хороші. Вибирайте той, що підходить під ваші завдання, але завжди починайте з найпростіших: img і фону, а потім вже ускладнюйте - якщо не вистачає.

**GraphQl , rest API , Soap**

REST (Representational State Transfer) — это архитектурный стиль, использующий стандартные HTTP методы и ресурсы для обмена данными между клиентом и сервером. Каждый ресурс в REST API идентифицируется уникальным URI (Uniform Resource Identifier).

#### **Основные характеристики**

* **Ресурс-ориентированность**: Вся информация представлена в виде ресурсов.
* **HTTP методы**:
  + GET — для получения данных.
  + POST — для создания новых данных.
  + PUT — для обновления существующих данных.
  + DELETE — для удаления данных.



### **Как работает REST API**

REST (Representational State Transfer) — это архитектурный стиль, использующий стандартные HTTP методы и ресурсы для обмена данными между клиентом и сервером. Каждый ресурс в REST API идентифицируется уникальным URI (Uniform Resource Identifier).

#### **Основные характеристики**

* **Ресурс-ориентированность**: Вся информация представлена в виде ресурсов.
* **HTTP методы**:
  + GET — для получения данных.
  + POST — для создания новых данных.
  + PUT — для обновления существующих данных.
  + DELETE — для удаления данных.
* **Статусные коды HTTP**: Используются для передачи состояния ответа (например, 200 OK, 404 Not Found, 500 Internal Server Error).

#### **Пример работы REST API**

1. **Запрос на получение данных (GET)**

http

Копировать код

GET /users/1 HTTP/1.1  
Host: api.example.com

**Ответ:**

json

Копировать код

{  
 "id": 1,  
 "name": "John Doe",  
 "email": "[john.doe@example.com](mailto:john.doe@example.com)"  
}

1. **Запрос на создание нового ресурса (POST)**

http

Копировать код

POST /users HTTP/1.1  
Host: api.example.com  
Content-Type: application/json  
  
{  
 "name": "Jane Doe",  
 "email": "[jane.doe@example.com](mailto:jane.doe@example.com)"  
}

**Ответ:**

json

Копировать код

{  
 "id": 2,  
 "name": "Jane Doe",  
 "email": "[jane.doe@example.com](mailto:jane.doe@example.com)"  
}

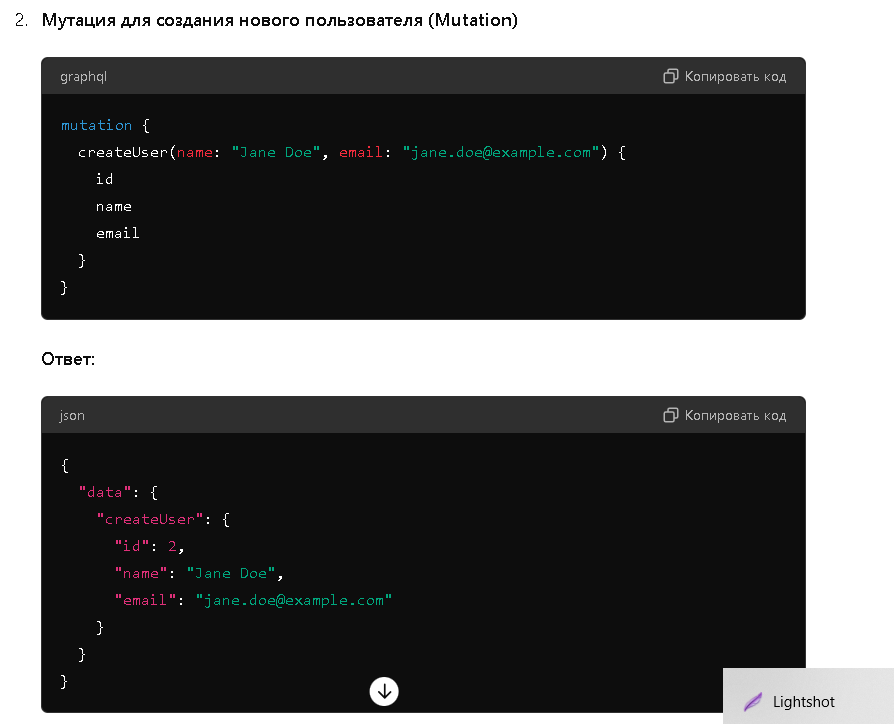
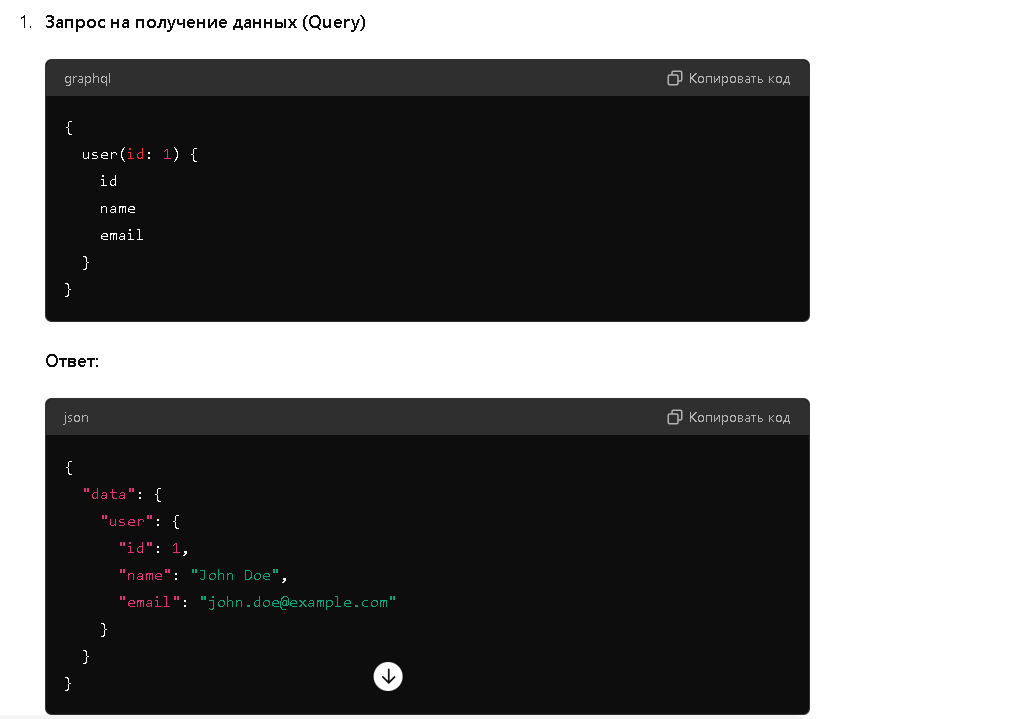
### **Как работает GraphQL**

GraphQL — это язык запросов для API, разработанный Facebook. Он позволяет клиентам точно указывать, какие данные им нужны, и получать их в одном запросе.

#### **Основные характеристики**

* **Запросы и мутации**:
  + **Запросы (queries)** — для получения данных.
  + **Мутации (mutations)** — для изменения данных.
* **Схема данных**: Описывает структуру данных, доступных через API.
* **Типизация**: Все данные имеют явные типы.

#### **Пример работы GraphQL**



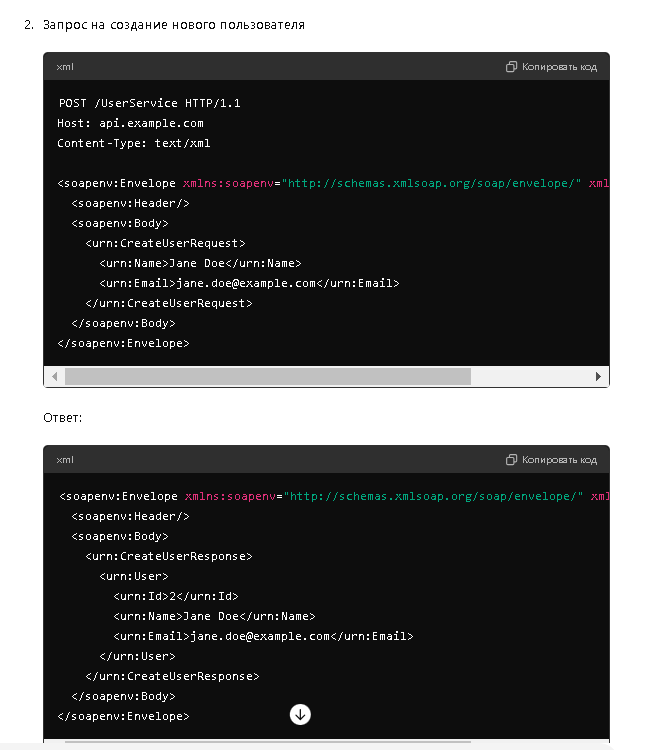
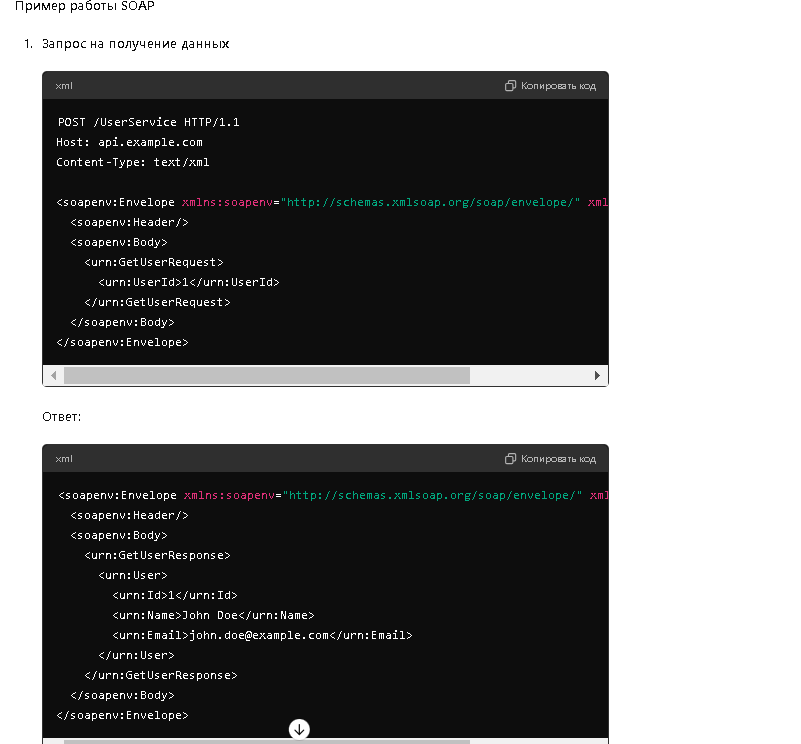
### **Как работает SOAP**

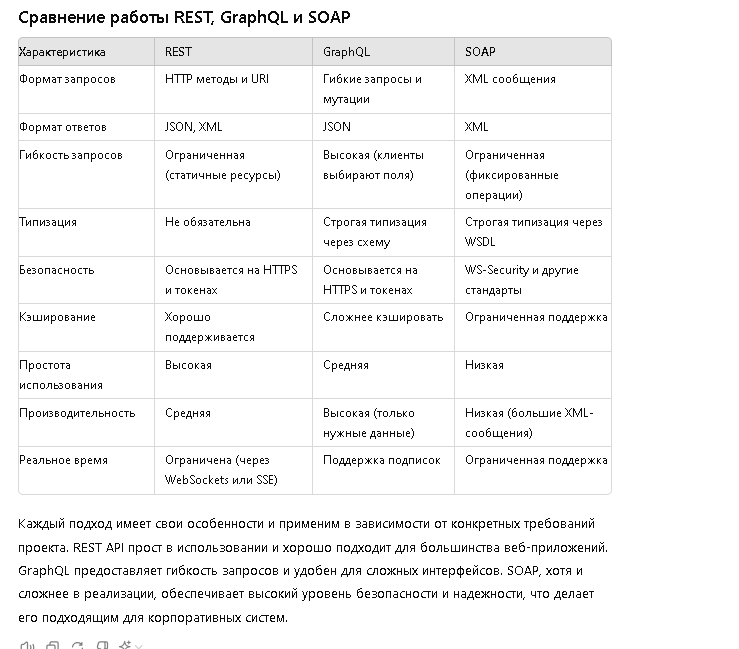
SOAP (Simple Object Access Protocol) — это протокол обмена сообщениями, использующий XML для форматирования запросов и ответов. Он часто используется в корпоративных приложениях для обеспечения высокого уровня безопасности и надежности.

#### **Основные характеристики**

* **Сообщения XML**: Форматирует данные в виде XML.
* **Транспорт**: Может использовать различные транспортные протоколы (например, HTTP, SMTP).
* *WS- стандарты*\*: Поддерживает множество стандартов для безопасности, транзакций и т.д.

#### **Пример работы SOAP**





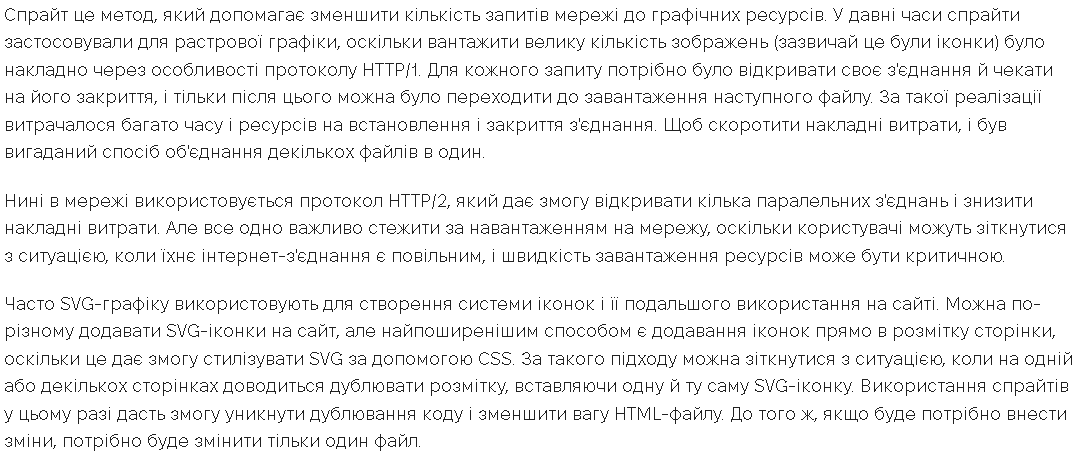
**Css анимации**

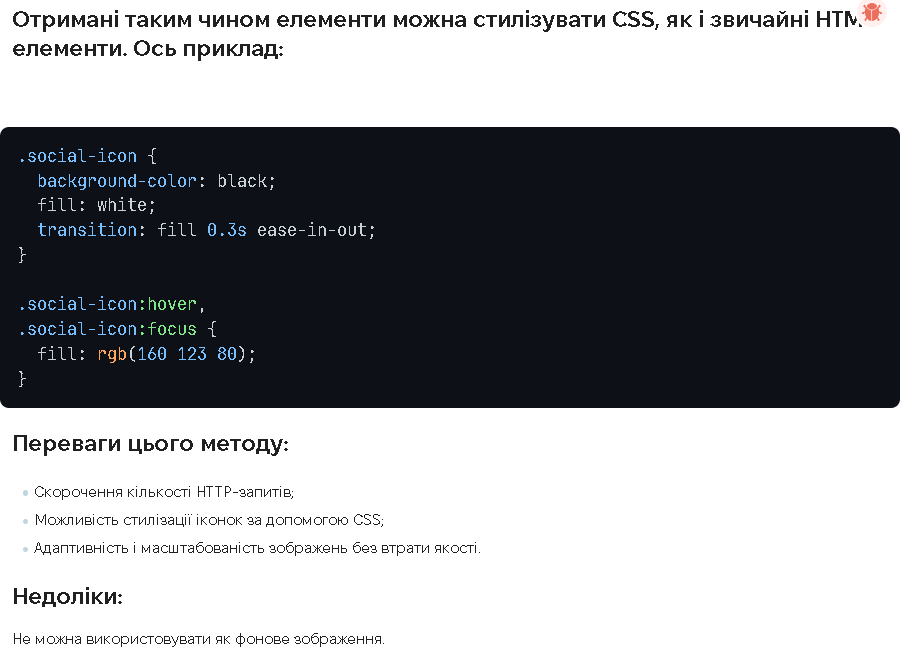
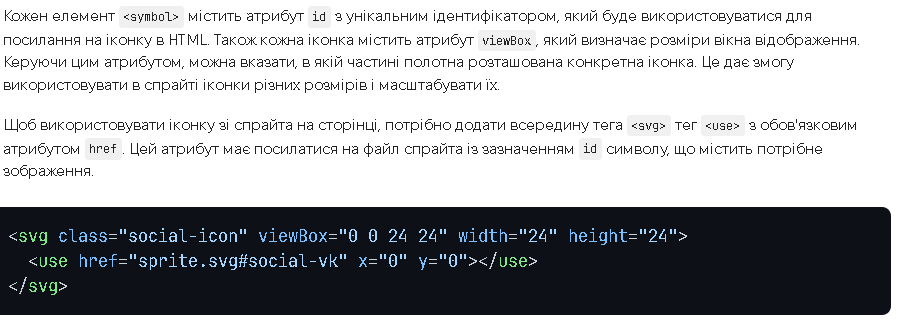
Свойство transition в CSS используется для управления переходами между состояниями элементов. Оно позволяет плавно изменять значения свойств в течение заданного времени. Вот все значения, которые могут быть использованы с transition:

1. **transition-property**: Определяет CSS-свойство(а), к которому(ым) применяется переход.
   * Значения:
     + none - Без перехода.
     + all - Все свойства.
     + Определенные свойства, такие как width, height, background-color, и т.д.
2. **transition-duration**: Указывает продолжительность перехода.
   * Значения:
     + Время в секундах (s) или миллисекундах (ms), например, 2s, 500ms.
3. **transition-timing-function**: Определяет скорость изменения свойств во время перехода.
   * Значения:
     + ease - Переход начинается медленно, затем ускоряется и снова замедляется.
     + linear - Равномерная скорость.
     + ease-in - Переход начинается медленно.
     + ease-out - Переход заканчивается медленно.
     + ease-in-out - Переход начинается и заканчивается медленно.
     + Кубические Безье-кривые, например, cubic-bezier(0.25, 0.1, 0.25, 1.0).
     + Шаги, например, steps(4, end).
4. **transition-delay**: Указывает задержку перед началом перехода.
   * Значения:
     + Время в секундах (s) или миллисекундах (ms), например, 1s, 200ms.

transition: [transition-property] [transition-duration] [transition-timing-function] [transition-delay];

### **Що таке SVG-спрайт?**



Использование SVG-спрайтов для веб-графики может значительно уменьшить нагрузку на HTTP и повысить производительность веб-сайта. Вот несколько причин, почему SVG-спрайты меньше нагружают HTTP:

### **1. Меньшее количество HTTP-запросов**

Когда вы используете отдельные изображения для каждой иконки или графики, каждый файл требует отдельного HTTP-запроса для загрузки. Это может значительно замедлить загрузку страницы, особенно если изображений много. SVG-спрайт представляет собой один файл, содержащий все необходимые изображения, что значительно уменьшает количество HTTP-запросов.

### **2. Быстрая загрузка**

SVG-спрайты позволяют загрузить все иконки и графику одним запросом. Это особенно полезно при использовании HTTP/1.1, где каждый запрос требует установки отдельного соединения. Даже при использовании HTTP/2, где соединения более эффективны, уменьшение количества запросов все равно положительно сказывается на производительности.

### **3. Кэширование**

SVG-спрайт загружается один раз и может быть закэширован браузером. При последующих загрузках страницы кэшированная версия спрайта будет использоваться, что сокращает время загрузки и уменьшает нагрузку на сервер.

### **4. Снижение времени загрузки страницы**

Меньшее количество HTTP-запросов и возможность кэширования приводят к более быстрой загрузке страницы. Это улучшает пользовательский опыт, так как пользователи не ждут загрузки множества отдельных изображений.

### **5. Меньший размер файлов**

SVG-файлы, как правило, имеют меньший размер по сравнению с растровыми изображениями (PNG, JPEG) при той же сложности графики. Это уменьшает объем передаваемых данных, что снижает нагрузку на сеть и ускоряет загрузку страницы.

### **Основные Свойства Анимации**

1. **animation-name**
2. Задает имя ключевых кадров анимации. Это имя должно соответствовать имени, указанному в @keyframes.

css

Копировать код

animation-name: myAnimation;

1. **animation-duration**

Определяет, как долго будет длиться анимация. Время указывается в секундах (s) или миллисекундах (ms).

css

Копировать код

animation-duration: 2s;

1. **animation-timing-function**

Определяет функцию времени для анимации, которая описывает, как скорость анимации изменяется во времени. Популярные функции включают:

* + linear: Равномерная скорость анимации.
  + ease: Начало и конец анимации медленные, середина быстрая.
  + ease-in: Начало анимации медленное.
  + ease-out: Конец анимации медленный.
  + ease-in-out: Начало и конец анимации медленные.
  + steps(): Анимация происходит в ступенях (резкие изменения).

css

Копировать код

animation-timing-function: ease-in-out;

1. **animation-delay**

Задает задержку перед началом анимации. Время указывается в секундах (s) или миллисекундах (ms).

css

Копировать код

animation-delay: 1s;

1. **animation-iteration-count**

Определяет количество раз, которое анимация будет повторяться. Может быть:

* + Числом, указывающим количество повторений.
  + infinite, чтобы анимация повторялась бесконечно.

css

Копировать код

animation-iteration-count: infinite;

1. **animation-direction**

Определяет направление анимации. Возможные значения:

* + normal: Анимация воспроизводится от начала до конца (по умолчанию).
  + reverse: Анимация воспроизводится от конца к началу.
  + alternate: Анимация воспроизводится от начала к концу и обратно.
  + alternate-reverse: Анимация воспроизводится от конца к началу и обратно.

css

Копировать код

animation-direction: alternate;

1. **animation-fill-mode**

Определяет, как стили элемента будут изменяться до начала и после окончания анимации. Возможные значения:

* + none: Стили не сохраняются (по умолчанию).
  + forwards: Стили сохраняются в конечном состоянии анимации.
  + backwards: Стили применяются до начала анимации.
  + both: Стили применяются до начала и сохраняются после окончания анимации.

css

Копировать код

animation-fill-mode: forwards;

1. **animation-play-state**

Определяет, запущена ли анимация или приостановлена. Возможные значения:

* + running: Анимация запущена (по умолчанию).
  + paused: Анимация приостановлена.

css

Копировать код

animation-play-state: paused;

