Ты привёл список стандартных **HTTP-статусов** и соответствующих классов из **Django**. Вот краткое пояснение каждого:

**🔄 1. 304 Not Modified**

* Кэш: означает, что у клиента уже есть актуальная версия ресурса
* Сервер не отправляет тело ответа

**❌ 2. 400 Bad Request**

* Неверный запрос от клиента (неправильный JSON, отсутствующие поля и т.д.)
* Используется для валидации

**🚫 3. 403 Forbidden**

* Доступ запрещён, **даже если авторизован**
* Например, нет прав на удаление ресурса

**📭 4. 404 Not Found**

* Ресурс не найден на сервере
* Классика — неправильный URL или ID

**🔒 5. 405 Method Not Allowed**

* Запрещённый HTTP-метод
* Например, POST на endpoint, который принимает только GET

**⛔ 6. 410 Gone**

* Ресурс **раньше существовал**, но теперь удалён **намеренно**
* Браузеры и поисковики воспринимают это как "навсегда удалено"

**💥 7. 500 Internal Server Error**

* Ошибка на сервере: баг, исключение, сбой в базе
* Всегда нужно логировать

📌 В Django можно вернуть любой из этих статусов с помощью соответствующего класса:

from django.http import HttpResponseNotFound, HttpResponseBadRequest

def my\_view(request):

return HttpResponseNotFound("Страница не найдена")

Хочешь — могу прислать таблицу с полным списком HTTP-статусов.

Тип Content-Type (заголовок HTTP) сообщает, **какой формат данных** передаётся в теле запроса или ответа. Вот самые популярные значения:

**📄 1. text/plain**

Простой текст без форматирования.

**🌐 2. text/html**

HTML-документ. Браузер отрисует как веб-страницу.

**🧾 3. application/json**

Формат JSON — часто используется в API.

Content-Type: application/json

**🧮 4. application/x-www-form-urlencoded**

Формат формы (key=value&key2=value2) — старый, но до сих пор используется с <form>.

**📎 5. multipart/form-data**

Передача файлов (например, изображений) через формы. Используется при enctype="multipart/form-data".

**🖼️ 6. image/png, image/jpeg, image/webp**

Изображения.

**📄 7. application/pdf**

PDF-документы.

**🎧 8. audio/mpeg, video/mp4**

Аудио и видео.

**🧠 9. application/octet-stream**

Бинарные данные — используется, если точный тип неизвестен.

💡 **На практике:**

* Для API → application/json
* Для отправки файла → multipart/form-data
* Для обычных форм → application/x-www-form-urlencoded

Хочешь — могу прислать список Content-Type заголовков по категориям (медиа, формы, API).

Отличный вопрос. Разнообразие хранилищ в браузере появилось потому, что **каждое из них решает разные задачи**, с разными ограничениями, сроками жизни, и уровнями доступа. Вот краткое сравнение:

**🍪 1. Cookies**

* 📦 Размер: до ~4 КБ
* 🕒 Жизнь: можно задать вручную (сессионные или постоянные)
* 📤 Отправляются **с каждым HTTP-запросом** на сервер
* 🛡 Часто используются для **аутентификации**, **трекинга**
* 👎 Медленные, небезопасные без HTTPS, легко "засоряются"

**💾 2. Local Storage**

* 📦 Размер: до 5–10 МБ
* 🕒 Жизнь: **пока пользователь не удалит вручную**
* 📥 Только на стороне клиента (не отправляется на сервер)
* ✅ Просто использовать: localStorage.setItem(...)
* 👎 Не поддерживает вложенные структуры (нужно JSON.stringify)

**🧭 3. Session Storage**

* 📦 Размер: ~5 МБ
* 🕒 Жизнь: **пока открыт вкладка/окно**
* 👯 Не разделяется между вкладками
* ✅ Удобно для временных данных, например, формы

**🗃 4. IndexedDB**

* 📦 Размер: сотни МБ (зависит от браузера)
* 🔍 База данных в браузере: поддерживает **индексацию, запросы, транзакции**
* 🧠 Идеально для сложных офлайн-приложений
* 👎 Сложный API (но можно обернуть библиотекой, например, idb)

**📦 5. Cache Storage (Service Worker)**

* 🎯 Используется для кэширования **HTTP-ответов**, файлов
* 🛠 Управляется через service worker
* 🔁 Позволяет офлайн-доступ к сайту
* ✅ Не для данных, а именно для **ресурсов (HTML, JS, картинки)**

**💡 Как выбирать:**

| **Задача** | **Рекомендуемое хранилище** |
| --- | --- |
| Кратковременные данные на вкладке | sessionStorage |
| Персональные настройки пользователя | localStorage |
| Данные между клиентом и сервером (например, auth) | cookies |
| Работа офлайн, большие объёмы, запросы | IndexedDB |
| Кэш ресурсов сайта | Cache Storage |

Хочешь — могу сделать таблицу сравнения с плюсами и минусами каждого хранилища.