🧠 WEB SCRAPING CHEATSHEET OF STAGE 1 AND STAGE 2

(Stage 0 + Stage 1 — From Zero to Sniper)

🔴 STAGE 0 — THE ANATOMY ("The X-Ray")

🎯 Goal

> **Before coding, I know exactly what to hit.**

Meaning:

- I don't blindly scrape HTML
- I first understand **where data comes from**

🧠 Mental Model (MOST IMPORTANT)

Website = 3 parts

- **Server** → actual data lives here
- **Browser** → middleman
- **JavaScript** → messenger (sometimes)

🔁 Two types of websites

1️⃣ Static / SSR (Server-Side Rendered)

- Page reload → data already there

- HTML contains the data
- Example: `quotes.toscrape.com`

👉 **requests + BeautifulSoup enough**

2️⃣ Dynamic / CSR (Client-Side Rendered)

- Page loads → empty / partial
- Scroll / click → data appears
- JavaScript fetches data in background
- Example: IMDb

👉 **Network tab is king**

🌐 Browser–Server Communication (Core Truth)

> **Browser server se background me baat karta rehta hai.**

Industry names:

- **XHR / Fetch** → background requests
- **JSON** → machine-friendly data
- **GraphQL** → “jo maango wahi mile” API system

🔍 DevTools — What to look at

Network tab (PRIMARY)

- **Doc** → HTML
- **Fetch / XHR** → real data
- Preview tab → structured data (keys, values)

Elements tab

- HTML structure (tree)
- Parent / child relationship

🪪 Headers & Cookies (Identity)

Headers

> **Who am I?**

- `User-Agent` → browser identity
- `Accept-Language` → language preference

➡️ Missing / fake headers → **403**

Cookies

> **Have I been here before?**

- Session / stamp
- Proof of continuity

🧪 Status Codes (Scraper POV)

Code	Meaning	Problem Type
200	OK	All good
403	Forbidden	**Identity problem** (headers)
429	Too many requests	**Speed problem** (rate)

🧠 Golden rule:

> **403 = WHO are you?**
> **429 = HOW FAST are you?**

🔥 Killer Move (Stage 0)

- Network → right click request → **Copy as cURL**
- Realization:
  > “Browser sirf middleman tha”

✅ Stage-0 unlocked when you can say:

> **“Data kahan se aa raha hai, mujhe pata hai.”**

🟢 STAGE 1 — THE SNIPER (STATIC EXTRACTION)

🎯 Goal

> **Build a scraper that survives small UI changes.**

Not a toy.
A **production mindset**.

🧰 Tools (WHY)

`requests`

- Python ka browser
- HTML fetch karta hai

`BeautifulSoup`

- HTML reader
- Tree ke andar navigate karta hai

📦 Analogy:

- `requests` = courier
- `BeautifulSoup` = unpack + read

🌳 HTML = TREE (THIS IS EVERYTHING)

Core rule

> **HTML ek tree hota hai, list nahi.**

- Parent → Child → Grandchild
- Jo element kisi aur ke andar likha ho → **child**

Example (quotes.toscrape)

```
div.quote        ← parent (ONE record)
├── span.text    ← child (quote text)
├── small.author ← grandchild
└── a.tag        ← grandchild
```

🧠 **One parent = one data record**

🎯 Scraping Strategy (Engineer Rule)

❌ Don’t grab text directly
✅ **First grab the parent**

Why?

- Parent gives **context**
- Context survives UI change

🎯 CSS SELECTORS (Sniper Logic)

Class vs ID

- `.class` → repeatable → data lists
- `#id` → unique → single element

Descendant vs Child

Selector	Meaning
`A B`	B is anywhere inside A
`A > B`	B is **direct child** of A

🔥 Selector Fragility (KILLER CONCEPT)

❌ Fragile (layout-based)

- `row`
- `col-md-8`
- `nth-child`

Breaks on redesign.

✅ Strong (meaning-based)

- `quote`
- `text`

- `author`
- `tags`

Rarely change.

🧠 Golden rule:

> **Never trust layout.**
> **Trust meaning.**

🧠 Unbreakable Selector Rules

1. Ignore layout classes
2. Grab parent first
3. Short + meaningful selector
4. Ask:
   > “Is this class meaning or design?”

🏁 STAGE 1 — MINDSET UNLOCKED

You now can:

- Read HTML as a tree
- Identify real data parents
- Write selectors that **survive**
- Predict what will break before it breaks

That’s **engineer-level thinking**.