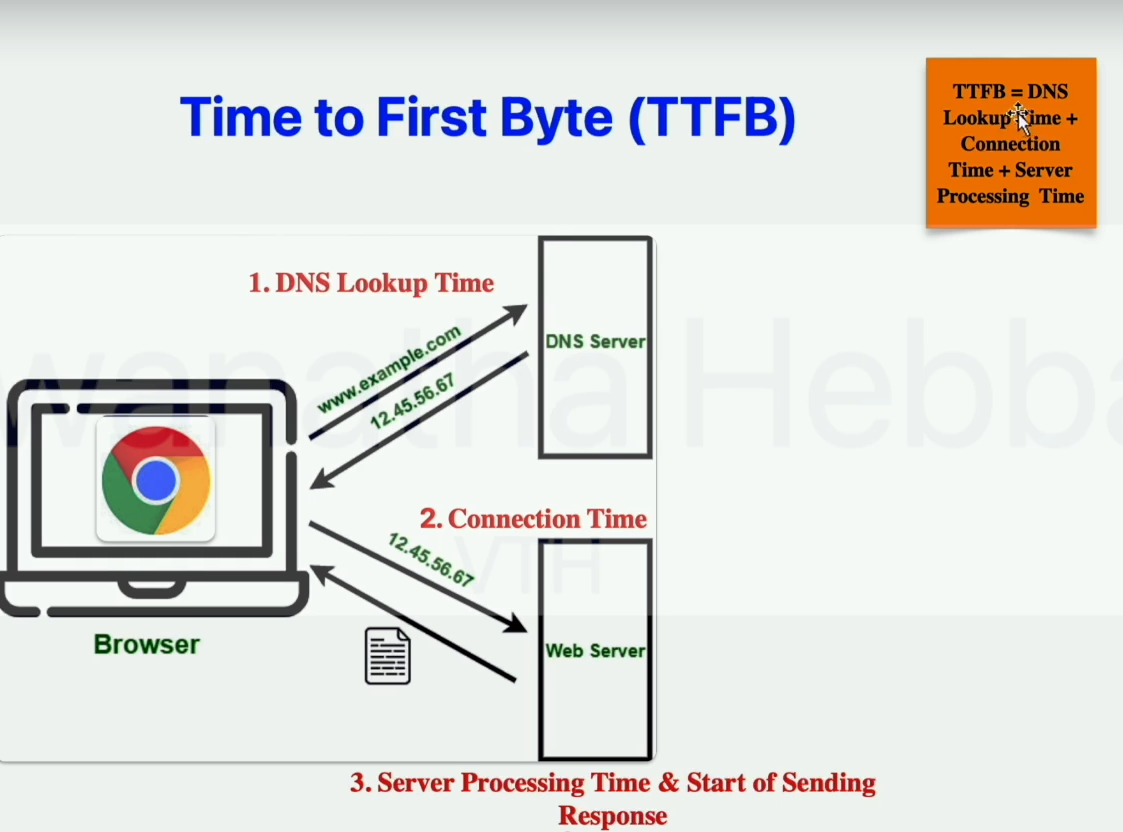
Time to First Byte (TTFB)

TTFB measures the time between when a user’s browser sends a request to the server and when it receives the **first byte** of the response. It's a critical web performance metric that reveals how responsive your server is.



🔍 **Why is TTFB Important?**

* 🚀 **Foundation for Performance**: A slow TTFB delays all other loading metrics like FCP and LCP.
* 👀 **User Experience**: Faster TTFB = quicker render = better engagement.
* 🔍 **SEO Relevance**: While not a Core Web Vital, TTFB affects overall page speed, influencing search rankings.

🧪 **What TTFB Includes**

1. 🔁 **Redirect Time**: Time spent following redirects (e.g., HTTP → HTTPS).
2. 🌐 **DNS Lookup**: Time to resolve the domain name to an IP address.
3. 🔐 **Connection & TLS Negotiation**: Establishing TCP and SSL handshakes.
4. 📤 **Request Sent**: Time to send the HTTP request.
5. 🧠 **Server Processing Time**: Backend logic, DB queries, and content generation.
6. 📥 **First Byte Received**: Time for the first byte to travel back to the browser.

📊 **What is a Good TTFB?**

| **Performance Level** | **TTFB Range** |
| --- | --- |
| ✅ Good | ≤ 800 ms |
| ⚠️ Needs Improvement | 800 ms – 1800 ms |
| ❌ Poor | > 1800 ms |

*Depends on whether the content is static or dynamic.*

⚠️ **Factors Affecting TTFB**

* 💻 **Hosting Quality**: Poor server resources = slow TTFB.
* 🗺️ **User-Server Distance**: Higher latency over greater physical distances.
* 🔎 **DNS Resolution Time**: Slow lookups = delayed response.
* 🔐 **TLS/SSL Handshake**: Adds security—but also latency.
* 🧾 **Backend Code & DB**: Inefficient code and queries slow down server processing.
* 🗃️ **Lack of Caching**: Every request is generated from scratch.
* 🔀 **Redirect Chains**: More redirects = more delay.
* 📈 **High Server Load**: Traffic spikes slow down processing.
* 🌐 **Third-Party API Calls**: External dependencies introduce latency.
* ❄️ **Serverless Cold Starts**: Startup delays in serverless functions.

⚙️ **How to Improve TTFB**

✅ **Choose a Better Hosting Plan**

* Opt for VPS, cloud hosting, or dedicated servers.

🌍 **Use a CDN**

* Offloads static content delivery, freeing your origin server for faster HTML responses.

🧠 **Implement Caching Mechanisms**

* **Server-side caching**: Store computed content.
* **Browser caching**: Avoid unnecessary re-downloads.
* **CDN caching**: Serve from edge locations.

🛠️ **Optimize Code and DB**

* Minimize logic complexity and query overhead.
* Refactor bloated code and eliminate redundancies.

🔀 **Minimize Redirects**

* Avoid unnecessary redirect chains and update links.

🌐 **Improve DNS**

* Use a premium DNS provider to reduce lookup delays.

📉 **Reduce Server Load**

* Monitor and scale resources based on demand.

⏩ **Stream HTML Markup**

* Send critical HTML chunks early (e.g., <head> section).

💼 **Utilize Service Workers**

* Cache application shells for faster repeat visits.

⚡ **Use 103 Early Hints**

* Let the browser preload assets before the full response arrives.

🧩 **Optimize CMS & Plugins**

* Remove unused plugins/themes and update frequently. Choose lightweight options.