HTML learning roadmap

Nice 🔥 now we're moving into **Performance & Optimization tags** in the `<head>` — this is the stuff that separates hobby projects from **production-grade apps**.

These don't change what the page *looks like*, but they **control speed, loading strategy, caching, and resource prioritization** → directly impacting **SEO, Core Web Vitals, and user experience**.

1️⃣ Viewport (mobile optimization)

```html
<meta name="viewport" content="width=device-width, initial-scale=1">
```

- Ensures responsive scaling on mobile.
- Without it → site zooms out (tiny text).
- ⚡ **Pro Tip**: Avoid `user-scalable=no` (bad for accessibility).

2️⃣ Resource Hints (Preload, Prefetch, Preconnect, DNS-Prefetch)

These tell the browser how to **prepare for resources before they're requested**.

🔹 Preload (highest priority)

```html
<link rel="preload" href="/fonts/roboto.woff2" as="font" type="font/woff2" crossorigin>
```

- Used for *critical* resources (fonts, hero image, above-the-fold CSS/JS).
- ⚡ Pro Tip: Always add `as="..."` so the browser knows how to handle it.

🔹 Prefetch (next-page optimization)

```html
<link rel="prefetch" href="/next-page.js" as="script">
```

- Low-priority fetch, used for **likely next-page navigation**.

🔹 **Preconnect (network handshake early)**

```
html

<link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
```

- Opens TCP/TLS handshake early for faster requests.

🔹 **DNS-Prefetch (just DNS lookup)**

```
html

<link rel="dns-prefetch" href="//cdn.example.com">
```

- Only resolves DNS; lighter than preconnect.

🚀 **Hidden Hack**: Use

```
html

<link rel="preconnect" href="https://fonts.googleapis.com" crossorigin>
<link rel="preload" href="/critical.css" as="style">
```

together → ensures **fastest font + CSS rendering**.

3️⃣ Async vs Defer (JS execution optimization)

```
html

<script src="main.js" async></script>
<script src="analytics.js" defer></script>
```

- `async` : loads in parallel, executes ASAP (good for ads/analytics).
- `defer` : loads in parallel, executes *after DOM ready* (good for app scripts).
- ⚡ **Pro Tip**: Always `defer` main app JS → avoids render blocking.

4️⃣ Preload Key Fonts (critical rendering path)

```
html

<link rel="preload" href="/fonts/open-sans.woff2" as="font" type="font/woff2" crossorigin>
```

- Prevents **FOIT/FOUT** (Flash of Invisible/Unstyled Text).
- Chrome Lighthouse audits complain if you don't do this.

5️⃣ HTTP Caching & Control via `<meta>`

```html
<meta http-equiv="Cache-Control" content="max-age=31536000, must-revalidate">
<meta http-equiv="Pragma" content="no-cache">
```

- Generally better set at **server level (HTTP headers)**, but meta works for client hints.
- ⚡ **Pro Tip**: For SPAs → use **service workers** instead of meta.

6️⃣ Content Security Policy (CSP) & Security-Performance Tradeoff

```html
<meta http-equiv="Content-Security-Policy" content="default-src 'self'; img-src https://cdn.example.com; script-src 'self' https://apis.google.com">
```

- Protects against XSS by **restricting sources**.
- Also improves performance by **blocking unnecessary requests**.

7️⃣ Other Optimization Tags

- **Theme color (for mobile browsers / PWA feel)**

```html
<meta name="theme-color" content="#0d6efd">
```

- **Manifest for PWA caching & offline**

```html
<link rel="manifest" href="/manifest.json">
```

- **Robots (control crawling)**

```html
<meta name="robots" content="index, follow">
```

8️⃣ Performance Priorities Summary (What to Always Do)

✅ Always:

- `<meta name="viewport">`
- `<script defer>`
- Preload critical fonts & CSS
- Preconnect to CDN/fonts

⚠️ Optional / Case-based:

- Prefetch (next-page heavy apps)
- Meta cache-control (if server headers unavailable)
- CSP (security + perf)

❌ Avoid:

- `user-scalable=no` → breaks accessibility.
- Blocking scripts in `<head>` without `defer/async`.

⚡ Reality Check:
Modern SEO (Google PageSpeed / Core Web Vitals) is **80% about these optimizations**, not just keywords.
Fast sites rank higher, convert better, and get fewer bounces.

👉 Do you want me to now **deep-dive into each resource hint (preload, preconnect, prefetch, dns-prefetch)** one by one with diagrams + real use-cases, or cover **async vs defer vs normal script loading** in the same bit-by-bit style?