# What Are the Key Performance Concepts?




Why do some websites feel snappy and responsive, while others feel sluggish? The answer lies in key performance concepts that affect how a page loads and renders. Understanding key web performance concepts is essential for building fast, smooth, and user-friendly websites.

Let's break down source order, the critical rendering path, latency, and more.

Let's look at source order first.

Source order refers to the way HTML elements are structured in the document. This determines what loads first and can significantly impact performance and accessibility.

Some best practices for this include:

- Placing critical content such as headings, navigation or main text higher in the HTML structure.
- Deferring non-essential scripts such as ones for analytics, or third-party widgets, so they don't block rendering.
- Using progressive enhancement, to ensure the core experience works even before styles and scripts load. Progressive enhancement is a way of building websites and applications based on the idea that you should make your page work with HTML first.

Here is an example of good source order, using the best practices we just went through.

```
<!-- Good source order: Essential content first -->
<h1>Welcome to FastSite!</h1>
<p>Critical information loads first.</p>
<script src="slow-script.js" defer></script>
```

By optimizing source order, we make sure users see important content as soon as possible.

Now let's look at critical rendering path.

The critical rendering path is the sequence of steps the browser follows to convert code into pixels on the screen.

Here are the key steps that we will go into later:

1. Parsing HTML: Builds the DOM (Document Object Model)
2. Parsing CSS: Builds the CSSOM (CSS Object Model)
3. JavaScript Execution: Can modify the DOM & CSSOM
4. Render Tree Construction: Combines the DOM & CSSOM
5. Layout & Painting: Determines element sizes & draws pixels
6. Optimizations:
    - Minimize render-blocking resources (e.g., large CSS files, unused JS).
    - Use async and defer attributes for scripts:
    - Load only essential styles first; defer non-critical CSS.

Overall, a shorter critical rendering path equals a faster perceived performance. We will go into this in more detail later on.

And finally, let's look at latency.

Latency is the time it takes for a request to travel between the browser and the server. So in other words, high latency equals slow pages.

Some ways of reducing latency include:

- Using CDNs, or in other words, Content Delivery Networks, to serve files from closer locations.
- Enabling compression using things such as Gzip to reduce file sizes.
- Optimizing images and using lazy loading - which we will also go into later.

By reducing latency, we make interactions feel instant.

So in conclusion, by optimizing source order, reducing the critical path, and cutting down latency, you can make your website feel fast.

## Questions

### What is source order?

The order in which a browser executes JavaScript.

The structure of a website's CSS files.

The way HTML elements are arranged in the document.

Correct!

The time it takes to load a webpage.

### How can you optimize the critical rendering path?

By adding more scripts to the `head` section.

By minimizing render-blocking resources and deferring scripts.

Correct!

By removing HTML elements.

By increasing latency.

### What is latency?

The delay in network requests.

Correct!

The process of rendering HTML and CSS.

The amount of images on a webpage.

The size of a website's JavaScript files.