*International Journal of Computational and Experimental Science and ENgineering (IJCESEN)*

*Vol. 10-No.4 (2024) pp. 4348-4360*
*http://www.ijcesen.com*

***ISSN: 2149-9144***



**Research Article**

# From FID to INP: Redefining Interaction Responsiveness and Its Impact on E-commerce Conversion in Large-Scale Digital Platforms

**Ranjith Reddy Gaddam***

Independent Researcher, India
* **Corresponding Author Email:** reachranjithgaddam@gmail.com- **ORCID:** 0009-0004-5857-096X

**Article Info:**

**DOI:** 10.22399/ijcesen.5218
**Received :** 15 May 2024
**Revised :** 28 June 2024
**Accepted :** 18 August 2024

**Keywords**

Interaction to Next Paint (INP),
Core Web Vitals,
Web Performance Optimization,
E-commerce Systems,
User Interaction Latency

**Abstract:**

The evaluation of web responsiveness has undergone a significant transformation with the shift from First Input Delay (FID) to Interaction to Next Paint (INP) as a primary performance metric. While FID provided a limited view by capturing only the latency of the initial user interaction, it failed to account for the sustained responsiveness required in modern, highly interactive web applications. INP addresses this limitation by measuring the latency of user interactions across the entire session and reporting a high-percentile value that reflects worst-case responsiveness.This paper presents a comprehensive technical analysis of the conceptual and mechanistic differences between FID and INP, emphasizing their implications for large-scale e-commerce platforms. It argues that INP introduces a fundamentally different optimization paradigm, requiring a holistic approach to the interaction processing pipeline, including input handling, JavaScript execution, rendering, and visual feedback. Through the synthesis of existing research and applied performance engineering principles, the study identifies key interaction patterns that are particularly susceptible to latency and proposes targeted architectural and implementation strategies to mitigate these issues.Furthermore, the paper examines the relationship between interaction responsiveness and e-commerce conversion outcomes, highlighting the role of latency in shaping user behavior and decision-making. The findings suggest that optimizing for INP is not merely a technical refinement but a strategic necessity for platforms seeking to maintain competitive performance and user engagement. By bridging measurement theory with practical engineering interventions, this work contributes to both academic discourse and industry practice in the domain of web performance optimization.

## 1. Introduction

Over the past decade, web performance has evolved from a backend-centric optimization concern into a central pillar of user experience design and digital business strategy. As web applications have grown increasingly complex—driven by client-side rendering frameworks, real-time data interactions, and rich media content—the expectations of users have shifted accordingly. Today's users demand instantaneous feedback to their actions, and even minor delays in responsiveness can significantly degrade perceived quality and trust (Souders, 2007; Nielsen, 1993). This expectation is particularly acute in high-stakes digital environments such as e-commerce platforms, where interaction speed directly influences purchasing decisions and revenue generation (Deloitte, 2020).

To standardize and quantify user experience on the web, Google introduced the Core Web Vitals framework, a set of metrics designed to measure real-world performance from a user-centric perspective (Google, 2023a). These metrics marked a departure from traditional load-based indicators by focusing instead on how quickly content becomes visible, how stable layouts remain during rendering, and how responsive a page is to user input. Among these, First Input Delay (FID) was initially adopted as a key measure of interactivity, capturing the delay between a user's first interaction and the browser's ability to begin processing it (Google, 2021).

While FID represented an important advancement in performance measurement, it quickly became evident that it did not fully capture the complexity of modern web interactions. By design, FID measures only the latency associated with the first

user input, disregarding all subsequent interactions that occur during a session. This limitation is increasingly problematic in contemporary web applications, where user engagement is defined not by a single interaction, but by a continuous sequence of actions. Empirical studies in human-computer interaction have long established that user perception of responsiveness is shaped by sustained interaction quality rather than isolated events (Seow, 2008). Consequently, a metric that captures only the initial delay provides an incomplete and potentially misleading representation of overall user experience (Belshe et al., 2015).

The inadequacy of FID becomes especially pronounced in the context of large-scale e-commerce platforms. Unlike static or content-driven websites, e-commerce systems are inherently interaction-intensive. Users engage in a variety of dynamic behaviors, including searching for products, applying filters, comparing options, updating cart contents, and navigating multi-step checkout processes. Each of these actions triggers a chain of frontend operations—event handling, state updates, rendering, and network communication—that collectively determine the responsiveness of the system. In such environments, performance bottlenecks can emerge at any point in the interaction lifecycle, making it insufficient to evaluate responsiveness based solely on the first input event (Akamai, 2017).

In response to these challenges, Interaction to Next Paint (INP) has been introduced as a more comprehensive metric for evaluating responsiveness. Unlike FID, INP considers the latency of all user interactions throughout the lifecycle of a page and reports a high-percentile value that reflects worst-case performance. This approach aligns more closely with real-world user perception, as users tend to remember delays that interrupt their workflow rather than those that occur during initial page load (Seow, 2008). By capturing the full interaction pipeline—from input to visual feedback—INP provides a more holistic and accurate assessment of responsiveness in modern web applications (Google, 2023b).

The transition from FID to INP signifies a broader shift in performance engineering philosophy. Under the FID paradigm, optimization efforts were often concentrated on ensuring that the browser's main thread was not blocked during the initial interaction. Techniques such as reducing JavaScript bundle size, deferring non-critical scripts, and optimizing resource loading were typically sufficient to achieve favorable FID scores. However, these strategies do not address the performance challenges that arise during ongoing interactions, such as long-running event handlers, inefficient rendering cycles, layout recalculations, and synchronous processing tasks (Grigorik, 2013).

INP, by contrast, requires a more systemic approach to performance optimization. Developers must now consider the entire interaction lifecycle, including how user inputs are processed, how application state is managed, and how updates are rendered to the screen. This shift places greater emphasis on architectural decisions, such as the adoption of asynchronous programming models, the use of concurrent rendering techniques, and the optimization of component-level updates in modern frontend frameworks (Tilkov & Vinoski, 2010; Facebook, 2022). It also necessitates more sophisticated monitoring and diagnostic tools capable of identifying performance bottlenecks at the level of individual interactions.

From a commercial perspective, the implications of this transition are substantial. Numerous studies have demonstrated a strong correlation between responsiveness and key business outcomes, including conversion rates, user engagement, and customer retention (Deloitte, 2020; Akamai, 2017). In e-commerce environments, where user decisions are often made rapidly and under cognitive load, delays in interaction feedback can disrupt decision-making processes and lead to abandonment. By providing a more accurate representation of real-world responsiveness, INP serves as a more reliable indicator of these outcomes, reinforcing its importance as both a technical and strategic metric.

This paper examines the transition from FID to INP through the lens of large-scale e-commerce systems, where the challenges and implications of this shift are most pronounced. It aims to provide a comprehensive analysis of the conceptual differences between the two metrics, the technical factors that influence interaction latency, and the engineering strategies required to optimize performance under the INP framework. By integrating insights from web performance research, frontend architecture, and user experience studies, the paper contributes to a deeper understanding of how responsiveness can be effectively measured and improved in modern web applications.

As the web continues to evolve toward increasingly interactive and dynamic paradigms, the ability to deliver consistently responsive experiences will become a defining factor in both user satisfaction and competitive advantage. The transition to INP reflects this evolution, establishing a new standard for what constitutes high-quality web performance in the contemporary digital landscape.

## 2. Background: Core Web Vitals and the FID Metric

The increasing complexity of modern web applications has necessitated a shift from system-centric performance metrics toward user-centric evaluation frameworks. In response, the Core Web Vitals initiative was introduced to standardize the measurement of real-world user experience by focusing on key dimensions such as loading performance, visual stability, and interactivity (Google, 2023a). Within this framework, First Input Delay (FID) emerged as the primary metric for quantifying responsiveness.

FID is formally defined as the time elapsed between a user's first interaction with a webpage—such as a click, tap, or keypress—and the point at which the browser is able to initiate processing of the corresponding event handler (Google, 2021). This delay typically occurs when the browser's main thread is occupied with tasks such as JavaScript execution, preventing immediate responsiveness. A threshold of less than 100 milliseconds has been widely accepted as indicative of satisfactory performance, consistent with established perceptual limits in human-computer interaction.

Despite its practical utility and ease of implementation, FID is fundamentally constrained by its narrow observational scope. Specifically, it captures only a single interaction event and is limited to the *input delay phase*, thereby excluding the subsequent stages of interaction processing, including computational execution and visual rendering. This partial measurement model introduces a disconnect between the metric and the actual user experience, particularly in applications characterized by sustained interaction sequences.

From a theoretical standpoint, user perception of responsiveness is inherently temporal and cumulative. Foundational research demonstrates that users evaluate system performance based on the continuity and predictability of feedback during task execution, rather than isolated response times (Nielsen, 1993; Seow, 2008). In this context, a metric that isolates the first interaction fails to account for latency experienced during subsequent, often more critical, interactions. Empirical studies further suggest that delays occurring during active engagement—such as form submission or content filtering—have a disproportionately negative impact on user satisfaction and task completion rates (Belshe et al., 2015).

These limitations become more pronounced in contemporary web architectures, where client-side rendering, asynchronous data fetching, and complex state management introduce multiple points of latency across the interaction lifecycle. A system may achieve an optimal FID score by ensuring that the main thread is unblocked during the initial interaction, yet still exhibit degraded responsiveness due to long-running tasks, inefficient rendering cycles, or layout recalculations triggered by subsequent inputs. This discrepancy highlights a critical gap between metric optimization and experiential quality.

To clarify the distinction between what FID measures and what it omits, Table 1 presents an analytical breakdown of its measurement scope and corresponding implications.

As illustrated, FID provides only a partial view of responsiveness, focusing narrowly on the initial interaction while neglecting the broader interaction pipeline. This limitation has significant implications for both performance engineering and user experience evaluation. In practice, it enables scenarios in which applications are technically optimized for FID yet remain perceptibly slow or unresponsive during actual use.

In summary, while FID represented a meaningful advancement in shifting performance measurement toward user-centric metrics, its restricted scope limits its effectiveness in capturing the complexities of modern web interaction. These shortcomings underscore the need for more comprehensive metrics capable of evaluating responsiveness across the full lifecycle of user engagement, thereby providing a more accurate and actionable representation of real-world performance.

## 3. Introducing INP: Measurement Model and Thresholds

The evolution of web applications toward highly interactive, client-driven architectures has exposed the limitations of traditional responsiveness metrics, necessitating a more comprehensive approach to performance evaluation. Interaction to Next Paint (INP) has been introduced as a user-centric metric designed to capture the latency of interactions across the entire lifecycle of a page session. Unlike earlier measures that focus on isolated events, INP provides a holistic representation of responsiveness by evaluating how efficiently a system reacts to user inputs throughout continuous engagement.

At its core, INP measures the elapsed time between a user interaction and the subsequent visual update that reflects the result of that interaction. Rather than relying on average latency, INP reports a high-percentile value—specifically the 98th percentile of all recorded interaction latencies—to approximate worst-case performance while filtering out extreme anomalies (Google, 2023c). This percentile-based approach aligns with established principles in human-computer interaction, which suggest that users are disproportionately influenced by the

slowest or most disruptive experiences during a session (Seow, 2008).

### 3.1 Interaction Lifecycle in INP

A key strength of INP lies in its ability to capture the complete interaction lifecycle. Each interaction is decomposed into three sequential phases, reflecting the underlying browser processing pipeline:

- Input Delay:
  This phase represents the time between the user's action (e.g., click, tap, or keypress) and the moment the browser begins executing the corresponding event handler. Delays in this phase typically arise from main-thread contention, where ongoing tasks prevent immediate response.
- Processing Time:
  This phase encompasses the execution of the event handler, including JavaScript computation, state updates, and any synchronous operations triggered by the interaction. Inefficient logic or long-running scripts can significantly increase latency at this stage.
- Presentation Delay:
  The final phase measures the time required for the browser to render and display the updated interface. This includes layout calculations, style recalculations, and painting operations necessary to produce the next visual frame.

By integrating these three components, INP captures not only when the system begins responding, but also when the user perceives the result of their interaction. This end-to-end measurement provides a more accurate reflection of perceived responsiveness compared to metrics that isolate individual phases.

### 3.2 Thresholds for Performance Evaluation

To facilitate consistent interpretation, INP defines standardized performance thresholds based on empirical research into user perception:

These thresholds are grounded in established usability principles, where delays below approximately 200 milliseconds are generally perceived as instantaneous, while delays exceeding this threshold begin to interrupt cognitive flow and reduce task efficiency (Nielsen, 1993).

### 3.3 Measurement Infrastructure

INP is implemented using the Event Timing API, which enables the browser to capture detailed timing information for individual user interactions, including their start and end points. Each interaction is logged with its associated latency, allowing for the computation of percentile-based metrics over the duration of a session.

In practice, INP data is collected and analyzed through both field and laboratory methods:

- Field Data (Real User Monitoring):
  Aggregated from real user sessions and reported through platforms such as the Chrome User Experience Report (CrUX), providing large-scale insights into real-world performance.
- Laboratory Data (Synthetic Testing):
  Measured using controlled tools such as Chrome DevTools and Lighthouse, enabling developers to simulate interactions and diagnose performance bottlenecks under reproducible conditions.

The combination of these approaches ensures both ecological validity and diagnostic precision, allowing developers to identify and address interaction latency at a granular level. As demonstrated in prior web performance research, the integration of real-user monitoring with controlled experimentation is essential for achieving sustained improvements in responsiveness (Wang et al., 2014).

### 3.4 Significance of the INP Model

By capturing the full interaction lifecycle and emphasizing worst-case latency, INP represents a significant advancement in performance measurement. It shifts the focus from isolated responsiveness to continuous interaction quality, aligning more closely with how users perceive and evaluate digital systems. This makes INP particularly relevant for modern applications characterized by dynamic interfaces and frequent user input, where performance bottlenecks can emerge at multiple stages of the interaction pipeline.

In summary, INP provides a comprehensive and actionable framework for evaluating responsiveness in contemporary web environments. Its measurement model not only addresses the shortcomings of earlier metrics but also establishes a more rigorous standard for optimizing user experience in interaction-intensive systems.

## 4. Mechanistic Differences Between FID and INP

The transition from First Input Delay (FID) to Interaction to Next Paint (INP) reflects a fundamental shift in how web responsiveness is

conceptualized and measured. While both metrics aim to quantify user interaction latency, they differ significantly in scope, measurement depth, and their implications for performance engineering. Understanding these mechanistic differences is essential for interpreting performance data and designing effective optimization strategies.

### 4.1 Limited Scope vs. Holistic Measurement

The most fundamental distinction between FID and INP lies in their scope of observation. FID is inherently limited, as it measures only the delay associated with the *first* user interaction. This narrow focus assumes that the initial interaction is representative of the overall user experience. However, in modern web applications—particularly those with rich, interactive interfaces—user engagement is characterized by a sequence of interactions rather than a single event.

In contrast, INP adopts a holistic measurement model by evaluating the latency of all interactions within a page session and reporting a high-percentile value that reflects near worst-case performance (Google, 2023b). This approach aligns more closely with empirical findings in usability research, which indicate that users tend to judge system responsiveness based on the slowest or most disruptive interactions encountered during their session (Seow, 2008). As a result, INP provides a more representative and reliable measure of real-world responsiveness.

### 4.2 Pipeline Coverage

A second critical difference lies in the depth of the interaction pipeline captured by each metric. FID measures only the input delay phase, i.e., the time between user input and the start of event handler execution. While this phase is important, it represents only a fraction of the total interaction lifecycle.

INP, by contrast, encompasses the entire interaction pipeline, including:

- JavaScript Execution:
  The processing of event handlers, state updates, and business logic triggered by user input.
- Layout Recalculation:
  The computation of element positions and dimensions, often required when the DOM is modified.
- Rendering and Painting:
  The final stages of updating the visual interface, including style application and pixel rendering.

By capturing these additional phases, INP accounts for delays that occur *after* the initial response begins—delays that are often most visible and impactful from the user's perspective. This comprehensive coverage ensures that performance issues related to rendering inefficiencies or computational overhead are not overlooked.

### 4.3 Architectural Implications

The differences in scope and measurement depth between FID and INP have significant implications for frontend architecture and performance optimization. Under the FID paradigm, it was often sufficient to ensure that the main thread was unblocked during the initial interaction, allowing the browser to respond quickly to the first input. However, this approach does not guarantee sustained responsiveness throughout the session.

In practice, applications that achieve favorable FID scores may still perform poorly under INP due to a range of architectural and implementation factors:

- Long JavaScript Tasks:

Extended execution of JavaScript can block the main thread, delaying both input handling and subsequent rendering operations. The Long Tasks API highlights how tasks exceeding 50 milliseconds can degrade responsiveness (Bocchi et al., 2016).

- Framework-Induced Re-renders:

Modern frontend frameworks, such as React, often trigger component re-renders in response to state changes. Without proper optimization, these re-renders can introduce significant processing overhead and delay visual updates (Facebook, 2022).

- Layout Thrashing:

Frequent and uncoordinated reads and writes to the DOM can force repeated layout recalculations, increasing rendering time and degrading performance (Grigorik, 2013).

- Synchronous Operations:

Blocking APIs and synchronous data processing can stall the interaction pipeline, preventing timely execution of event handlers and delaying visual feedback (Tilkov & Vinoski, 2010).

These factors illustrate that responsiveness is not determined solely by the initial availability of the main thread, but by the efficiency of the entire interaction pipeline. Consequently, optimizing for INP requires a systemic approach that addresses performance at multiple levels, including event handling, state management, rendering strategies, and architectural design.

### 4.4 From Isolated Optimization to Systemic Performance Engineering

The shift from FID to INP necessitates a corresponding shift in optimization philosophy. Rather than focusing on isolated improvements—such as reducing initial JavaScript execution—developers must adopt a holistic performance engineering approach that ensures consistent responsiveness across all user interactions.

This involves:

- Minimizing main-thread blocking across the entire session
- Structuring applications to support asynchronous and incremental updates
- Reducing unnecessary re-renders and layout recalculations
- Optimizing rendering pipelines for faster visual feedback

In this context, INP serves not only as a measurement metric but also as a design constraint, guiding the development of systems that prioritize sustained interactivity and user-centric performance.

In summary, the mechanistic differences between FID and INP extend beyond measurement methodology to fundamentally reshape how responsiveness is understood and optimized. While FID provides a limited snapshot of initial interaction latency, INP offers a comprehensive view of the interaction lifecycle, revealing performance issues that would otherwise remain undetected. This shift underscores the need for more sophisticated and integrated approaches to performance optimization in modern web applications.

## 5. E-commerce Interaction Patterns and INP Risk Zones

E-commerce platforms represent one of the most interaction-intensive categories of modern web applications. Unlike content-driven websites, where user engagement may be limited to navigation and passive consumption, e-commerce systems require users to perform a sequence of active, decision-driven interactions. These include searching for products, applying filters, selecting variants, updating cart contents, and completing multi-step checkout processes. Each of these actions triggers a chain of frontend operations—event handling, state updates, rendering, and network communication—that directly influence interaction latency.

From an INP perspective, such environments are particularly susceptible to performance degradation because latency is not confined to a single interaction but accumulates across multiple user actions. Certain interaction patterns consistently emerge as high-risk zones, where the likelihood of elevated interaction latency is significantly greater due to architectural and computational complexity.

### 5.1 Search Filtering

Search and filtering functionalities are central to the e-commerce user journey, enabling users to refine product listings based on attributes such as price, category, brand, and availability. These interactions are typically implemented using dynamic, client-side logic that updates the visible product set in real time.

However, filtering operations often trigger multiple simultaneous processes, including state updates, API requests, and DOM re-rendering. In large product catalogs, these updates can result in significant computational overhead, especially when filtering logic is executed synchronously. Additionally, frequent DOM manipulations may lead to layout recalculations and repaint operations, contributing to both processing and presentation delays. As a result, search filtering is a common source of long tasks that negatively impact INP.

### 5.2 Add-to-Cart Operations

The add-to-cart interaction is a critical conversion point in the e-commerce funnel and typically involves a combination of frontend and backend processes. Upon user input, the system may perform synchronous validation (e.g., stock availability, variant selection), initiate API calls to update server-side cart data, and trigger UI updates to reflect the new cart state.

These operations are often tightly coupled and executed in a blocking manner, increasing the processing time component of interaction latency. If not properly optimized, such interactions can delay visual feedback, creating a perception of sluggishness or uncertainty for the user. Given the importance of this action in the purchase journey, any latency at this stage can have a direct impact on user trust and conversion rates.

### 5.3 Product Variant Selection

Product variant selection—such as choosing size, color, or configuration—is another interaction pattern with significant INP implications. These interactions frequently trigger updates to product images, pricing information, and availability indicators.

In many implementations, variant selection leads to image reloading, dynamic content replacement, and recalculation of layout structures. If these updates are not efficiently managed, they can introduce delays in both processing and rendering phases. For example, large image assets or unoptimized media loading strategies can increase presentation delay,

while complex state dependencies may prolong processing time. Consequently, variant selection interactions often exhibit noticeable latency under suboptimal conditions.

### 5.4 Checkout Flows

Checkout processes represent the most complex and latency-sensitive phase of the e-commerce experience. These workflows typically involve multiple steps, including form input, validation, address verification, payment processing, and order confirmation.

Each step introduces potential performance bottlenecks. Client-side validation logic, especially when executed synchronously, can delay input responsiveness. Additionally, interactions with external payment gateways and APIs may introduce network-induced latency. Research has shown that delays during checkout significantly increase the likelihood of cart abandonment, as users are particularly sensitive to disruptions during transactional interactions (Deloitte, 2020).

From an INP perspective, checkout flows are critical because they involve repeated interactions under time-sensitive conditions. Any degradation in responsiveness during this phase can have a disproportionately negative impact on overall user experience and business outcomes.

### 5.5 Image Carousels and Interactive Media

Image carousels and interactive media components are widely used in e-commerce platforms to enhance product visualization. These components often rely on animations, transitions, and frequent updates to the Document Object Model (DOM).

While visually engaging, such interactions can be computationally expensive. Continuous updates to the DOM, combined with animation logic, can lead to presentation delays, particularly if rendering is not offloaded to the compositor thread. Inefficient animation techniques or excessive use of layout-affecting properties may further exacerbate latency by triggering repeated layout recalculations and paint operations.

As a result, image carousels represent a common source of performance issues, particularly on devices with limited processing power or in scenarios involving high-resolution media.

### 5.6 Synthesis of INP Risk Zones

The interaction patterns discussed above highlight a common theme: INP degradation is often driven by cumulative latency across multiple stages of the interaction pipeline. Whether caused by computational overhead, rendering inefficiencies, or synchronous processing, these delays manifest as slow or inconsistent feedback to user inputs.

In summary, e-commerce platforms inherently contain multiple interaction patterns that are prone to elevated interaction latency. These high-risk zones arise from the complexity of frontend operations and the need to process user inputs in real time. By identifying and analyzing these patterns, developers can prioritize optimization efforts toward the interactions that most significantly impact INP and, by extension, user experience and conversion performance.

## 6. Engineering Interventions for INP Improvement

Improving Interaction to Next Paint (INP) requires a shift from isolated optimizations toward a system-level performance strategy that addresses latency across the entire interaction pipeline. Given that INP captures input delay, processing time, and presentation delay, effective optimization must target all three phases. This section outlines key engineering interventions that have proven effective in reducing interaction latency in modern web applications, particularly in complex, interaction-heavy environments.

### 6.1 Breaking Up Long Tasks

One of the most significant contributors to poor INP performance is the presence of long-running JavaScript tasks that block the browser's main thread. When the main thread is occupied, it cannot process new user inputs, resulting in increased input delay and degraded responsiveness.

A practical approach to mitigating this issue is to split long tasks into smaller, asynchronous units using techniques such as setTimeout, requestIdleCallback, or scheduler-based yielding. By breaking execution into manageable chunks, the browser is given opportunities to handle user input between tasks, thereby improving responsiveness (Grigorik, 2013).

This approach is particularly effective in scenarios involving large data processing or complex computations triggered by user interactions, where uninterrupted execution would otherwise lead to noticeable delays.

### 6.2 React Concurrency Features

Modern frontend frameworks, such as React, provide built-in mechanisms to prioritize user interactions and manage rendering more efficiently. Features introduced in React 18, including

useTransition and useDeferredValue, enable developers to distinguish between urgent and non-urgent updates.

- useTransition allows non-critical updates to be deferred, ensuring that high-priority interactions—such as clicks or input events—are processed without delay.
- useDeferredValue enables the postponement of expensive computations or rendering tasks, reducing the likelihood of blocking the main thread during active user input.

By leveraging these concurrency features, developers can ensure that critical interactions receive immediate attention, thereby reducing both input delay and processing time (Facebook, 2022).

### 6.3 Event Handler Optimization

Event handlers are central to the interaction pipeline, and inefficient handler logic can significantly increase processing time. In many applications, event handlers perform multiple synchronous operations, including validation, state updates, and data transformations, all of which contribute to latency.

Optimizing event handlers involves:

- Minimizing synchronous computations
- Avoiding unnecessary state updates
- Deferring non-essential operations
- Reducing dependency chains that trigger cascading updates

By streamlining event handler logic, developers can reduce the time required to process user input, leading to faster interaction completion and improved INP scores.

### 6.4 Rendering Optimization

The presentation phase of interaction latency is heavily influenced by the efficiency of the browser's rendering pipeline. Poor rendering performance can lead to delays in visual feedback, even when input handling and processing are efficient.

Key strategies for rendering optimization include:

- CSS Containment: Restricting layout and style calculations to specific sections of the DOM, thereby reducing the scope of reflows.
- Compositor-Only Animations: Using properties such as transform and opacity that can be handled by the GPU, avoiding layout and paint operations.
- Reducing Layout Thrashing: Minimizing frequent reads and writes to layout properties that force recalculations.

These techniques help reduce presentation delay by ensuring that visual updates are rendered efficiently and with minimal computational overhead (Google, 2023d).

### 6.5 Server-Side Rendering (SSR) and Streaming

Server-Side Rendering (SSR) is an architectural approach that shifts part of the rendering workload from the client to the server. By delivering pre-rendered HTML to the browser, SSR reduces the amount of JavaScript that must be parsed and executed on the client side before the page becomes interactive.

Recent advancements, such as streaming SSR, further enhance this approach by allowing content to be progressively rendered and delivered to the client. This reduces initial load times and enables earlier interaction readiness, thereby decreasing the likelihood of main-thread blocking during early user interactions (Osmani & Stefanovsky, 2019).

By lowering the computational burden on the client, SSR contributes to improved responsiveness across both initial and subsequent interactions.

### 6.6 Micro-frontend Coordination

Large-scale applications increasingly adopt micro-frontend architectures, where different parts of the user interface are developed and deployed independently. While this approach enhances modularity and scalability, it can introduce performance challenges related to inter-module communication and coordination.

Unoptimized communication between micro frontends can lead to:

- Redundant rendering cycles
- Increased state synchronization overhead
- Cascading updates across components

To mitigate these issues, it is essential to establish efficient communication patterns, minimize shared state dependencies, and ensure that updates are localized whenever possible. Reducing coordination overhead helps prevent cascading delays that can negatively impact interaction latency (Newman, 2015).

### 6.7 Toward a Holistic Optimization Strategy

The interventions discussed above highlight that improving INP is not achieved through a single optimization technique but through a comprehensive approach that addresses all stages of the interaction pipeline. Effective INP optimization requires:

- Managing main-thread workload to reduce input delay

- Streamlining computation to minimize processing time
- Optimizing rendering to ensure rapid visual feedback

By integrating these strategies into the design and implementation of web applications, developers can achieve sustained improvements in responsiveness and deliver a more consistent and satisfying user experience.

In summary, INP optimization demands a shift toward holistic performance engineering, where responsiveness is treated as a system-wide property rather than a localized concern. The techniques outlined in this section provide a practical foundation for addressing interaction latency in modern web applications, particularly in complex, high-interaction environments such as e-commerce platforms.

## 7. Measurement and Monitoring Framework

Effective optimization of Interaction to Next Paint (INP) requires a robust measurement and monitoring framework that combines real-world user data, controlled laboratory testing, and fine-grained attribution techniques. Given that INP reflects user-perceived responsiveness across diverse devices, network conditions, and interaction patterns, relying on a single measurement approach is insufficient. Instead, a multi-layered strategy is essential for both accurate diagnosis and continuous performance improvement.

### 7.1 Field Data

Field data, also referred to as Real User Monitoring (RUM), captures performance metrics directly from actual user interactions in production environments. This data provides high ecological validity, reflecting real-world conditions such as device variability, network latency, and user behavior patterns.

Two primary sources are widely used for collecting INP field data:

- Chrome User Experience Report (CrUX): CrUX aggregates anonymized performance data from millions of users, providing large-scale insights into how websites perform across different regions, devices, and connection types. It enables benchmarking against industry standards and tracking performance trends over time.
- Web Vitals JavaScript Library: This library allows developers to collect INP and other Core Web Vitals metrics directly within their applications. It supports granular data collection at the page and interaction level, enabling organizations to correlate performance metrics with specific user journeys and business outcomes.

Field data is particularly valuable for identifying systemic performance issues that may not be apparent in controlled testing environments, making it a critical component of any INP monitoring strategy.

### 7.2 Lab Data

While field data provides real-world insights, laboratory (lab) data offers a controlled environment for diagnosing performance issues and testing optimizations. Lab tools simulate user interactions under predefined conditions, enabling reproducible analysis of performance bottlenecks.

Key tools for lab-based INP analysis include:

- Chrome DevTools Performance Panel: This tool provides detailed timelines of browser activity, including JavaScript execution, layout recalculations, and rendering operations. It allows developers to trace individual interactions and identify sources of latency within the interaction pipeline.
- Lighthouse Audits: Lighthouse provides automated performance audits, including simulated interaction metrics and recommendations for improvement. It is particularly useful for identifying common performance anti-patterns and validating optimization efforts.

Lab data complements field data by enabling precise, interaction-level debugging, making it an essential tool for iterative performance engineering.

### 7.3 Attribution and Bottleneck Identification

Beyond measurement, effective INP optimization requires the ability to attribute latency to specific causes within the interaction pipeline. Without such attribution, performance data remains descriptive rather than actionable.

The Long Animation Frames (LoAF) API provides a mechanism for identifying long frames that exceed expected rendering durations, enabling developers to pinpoint interactions that contribute disproportionately to INP (Google, 2024). By correlating long frames with specific JavaScript tasks, rendering operations, or layout recalculations, developers can isolate the root causes of latency and prioritize targeted interventions.

This attribution capability is particularly important in complex applications, where multiple subsystems—such as event handling, state

management, and rendering—interact to produce the final user experience.

### 7.4 Integrated Monitoring Strategy

An effective INP monitoring framework integrates:

- Field data for real-world performance insights
- Lab data for controlled diagnostics and testing
- Attribution tools for identifying and resolving bottlenecks

Such a framework enables continuous performance evaluation, ensuring that improvements are both measurable and sustainable over time.

## 8. Impact on E-commerce Conversion

Web performance has long been recognized as a critical factor influencing user behavior and business outcomes, particularly in e-commerce environments where user decisions are time-sensitive and interaction-driven. A substantial body of research demonstrates a direct relationship between responsiveness and key performance indicators such as conversion rate, user engagement, and revenue generation.

Empirical studies highlight the sensitivity of users to even minor delays in interaction:

- A delay of 100 milliseconds in responsiveness can lead to a reduction in conversion rates of up to 7%, underscoring the economic impact of latency (Akamai, 2017).
- Faster websites consistently exhibit higher levels of user engagement, including longer session durations and increased interaction frequency (Souders, 2007).
- Improvements in Core Web Vitals have been shown to correlate with measurable gains in revenue and user retention, reinforcing the business value of performance optimization (Google, 2020).

These findings are particularly relevant in the context of INP, which captures real interaction latency rather than isolated or synthetic performance indicators. Unlike earlier metrics, INP reflects the responsiveness experienced during critical user actions, such as product selection, cart updates, and checkout processes.

From a behavioral perspective, delays in interaction feedback disrupt cognitive flow and introduce uncertainty, increasing the likelihood of user frustration and abandonment. In e-commerce settings, where users often evaluate multiple options and make rapid decisions, such disruptions can significantly reduce conversion likelihood (Deloitte, 2020).

Moreover, INP's focus on worst-case interaction latency aligns with the observation that users tend to remember negative experiences more strongly than positive ones. A single slow interaction—particularly during a critical step such as checkout—can disproportionately influence overall perception and lead to session abandonment.

### 8.1 INP as a Predictor of Business Outcomes

By providing a comprehensive measure of interaction responsiveness, INP serves as a stronger predictor of user satisfaction and conversion performance compared to earlier metrics. Its ability to capture latency across the entire interaction lifecycle makes it particularly valuable for identifying performance issues that directly impact user decision-making.

For e-commerce platforms, optimizing INP is therefore not merely a technical objective but a strategic imperative. Improvements in INP can lead to:

- Increased conversion rates
- Higher user engagement
- Reduced bounce and abandonment rates
- Enhanced overall user satisfaction

In summary, the relationship between web performance and business outcomes is both well-established and highly significant. By capturing the latency of real user interactions, INP provides a more accurate and actionable metric for optimizing this relationship. For e-commerce platforms operating in competitive digital markets, prioritizing INP optimization is essential for achieving sustained growth and delivering high-quality user experiences.

## 9. Discussion

The transition from First Input Delay (FID) to Interaction to Next Paint (INP) reflects a broader and necessary shift toward user-centric performance evaluation in modern web systems. This shift is not merely incremental but represents a fundamental change in how responsiveness is conceptualized, measured, and optimized. While FID provided a limited snapshot of initial interactivity, it allowed developers to achieve acceptable performance scores through relatively narrow optimizations, such as reducing initial JavaScript execution or deferring non-critical resources. These approaches, although effective for improving FID, often failed to address the deeper structural inefficiencies that affect sustained interaction quality.

In contrast, INP introduces a more stringent and holistic performance requirement by capturing latency across the entire interaction lifecycle. As a result, it exposes performance bottlenecks that were previously obscured under the FID paradigm. This increased sensitivity necessitates deeper architectural interventions, moving beyond surface-level fixes toward systemic performance engineering.

A key implication of this transition is the growing importance of asynchronous design patterns. By decoupling computational tasks and enabling non-blocking execution, asynchronous architectures reduce main-thread contention and allow user interactions to be processed with minimal delay. This approach is particularly critical in applications with high interaction density, where synchronous operations can quickly accumulate and degrade responsiveness.

Equally important is the development of efficient rendering pipelines. Modern web applications must minimize unnecessary layout recalculations, reduce rendering overhead, and leverage browser capabilities such as GPU acceleration to ensure timely visual feedback. Techniques such as incremental rendering, virtualization, and compositor-only updates play a crucial role in achieving these objectives.

Another critical factor is the optimization of component hierarchies within frontend frameworks. Poorly structured component trees can lead to excessive re-renders and inefficient state propagation, increasing both processing time and presentation delay. By designing components with clear boundaries, minimizing shared state, and adopting selective rendering strategies, developers can significantly improve interaction performance.

These architectural considerations align closely with contemporary frontend paradigms that emphasize responsiveness, scalability, and maintainability. As web applications continue to grow in complexity, performance can no longer be treated as an afterthought; it must be embedded into the design and development process from the outset. In this context, INP serves not only as a measurement metric but also as a guiding principle for building systems that deliver consistent and reliable user experiences (Grigorik, 2013).

***Table 1:*** *Analytical Limitations of First Input Delay (FID)*

| Measurement Dimension | Captured by FID | Excluded from FID | Implication |
|---|---|---|---|
| Interaction Coverage | First interaction | All subsequent interactions | Does not represent session-wide responsiveness |
| Interaction Phases | Input delay | Processing time, rendering delay | Ignores latency after event handling begins |
| Temporal Continuity | Single event | Continuous interaction flow | Fails to capture consistency of responsiveness |
| Alignment with User Perception | Partial | Full experiential latency | Weak correlation with perceived performance |

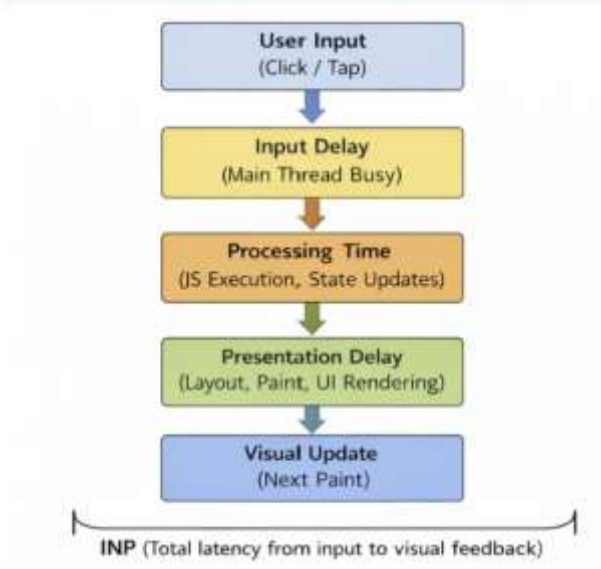


***Figure 1:****Interaction to Next Paint(INP) Measurement Pipeline*

***Table 2:*** *Thresholds*

| INP Value | Performance Category | User Experience Interpretation |
|---|---|---|
| < 200 ms | Good | Interactions feel immediate and highly responsive |
| 200–500 ms | Needs Improvement | Noticeable delay that may affect usability |
| > 500 ms | Poor | Significant lag likely to disrupt user interaction |

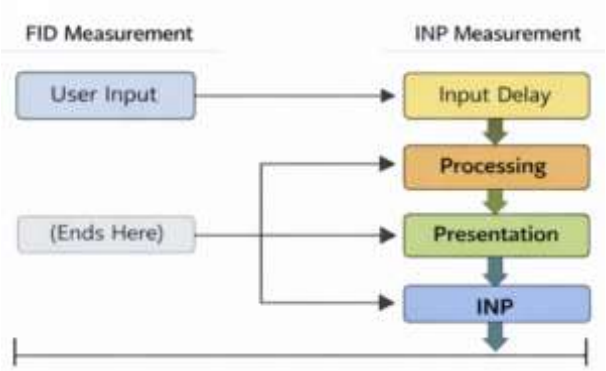


***Figure 2:*** *Comparison of FID Vs. INP measurement scope*

***Table 3****: High-Risk Interaction Patterns and INP Impact*

| Interaction Type | Primary Latency Source | INP Phase Affected | Risk Level |
|---|---|---|---|
| Search Filtering | State updates, DOM reflows | Processing, Presentation | High |
| Add-to-Cart | Validation, API calls, UI updates | Processing | High |
| Variant Selection | Image loading, layout recalculation | Processing, Presentation | Medium–High |
| Checkout Flows | Form validation, network latency | Input, Processing | High |
| Image Carousels | Animations, DOM updates | Presentation | Medium |

## 10. Conclusions and Future Directions

This study has examined the transition from FID to INP as a defining development in the evolution of web performance metrics. By moving beyond the limitations of initial interaction measurement, INP establishes a more comprehensive framework for evaluating responsiveness, capturing the full lifecycle of user interactions and emphasizing worst-case latency. This shift represents a fundamental redefinition of what constitutes responsive web performance.

For e-commerce platforms, the implications of this transition are particularly significant. Given their inherently interactive nature, these systems are highly sensitive to delays in user input processing and visual feedback. As demonstrated throughout this paper, performance bottlenecks can arise at multiple stages of the interaction pipeline, and addressing them requires a coordinated, system-level approach. Optimizing for INP is therefore not only a technical necessity but also a strategic priority for maintaining competitiveness in digital markets.Looking forward, several avenues for future research and development emerge:

- INP Optimization in Mobile-First Environments: Mobile devices, with their limited computational resources and variable network conditions, present unique challenges for maintaining low interaction latency. Investigating optimization strategies tailored to mobile contexts remains an important area of study.
- Integration with AI-Driven Interfaces: The increasing adoption of AI-powered features—such as personalized recommendations, conversational interfaces, and real-time analytics—introduces new sources of computational complexity. Understanding how these systems interact with INP and identifying strategies to mitigate their performance impact is a promising direction for future research.
- Interaction with Emerging Browser APIs: Advances in browser technologies, including new scheduling mechanisms, rendering APIs, and performance monitoring tools, offer opportunities to further enhance responsiveness. Exploring how these capabilities can be leveraged to optimize INP will be critical for the next generation of web applications.

In conclusion, INP provides a more accurate, comprehensive, and actionable framework for evaluating web responsiveness. By aligning performance measurement with real user experience, it enables developers and organizations to identify and address the factors that most

significantly impact usability and engagement. As the web continues to evolve, the adoption of INP-driven optimization strategies will play a central role in delivering high-quality digital experiences and achieving sustained business success.

## Author Statements:

- **Ethical approval:** The conducted research is not related to either human or animal use.
- **Conflict of interest:** The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper
- **Acknowledgement:** The authors declare that they have nobody or no-company to acknowledge.
- **Author contributions:** The authors declare that they have equal right on this paper.
- **Funding information:** The authors declare that there is no funding to be acknowledged.
- **Data availability statement:** The data that support the findings of this study are available on request from the corresponding author. The data are not publicly available due to privacy or ethical restrictions.
- **Use of AI Tools:** The author(s) declare that no generative AI or AI-assisted technologies were used in the writing process of this manuscript.

## References


[1] Deloitte. (2020). *Milliseconds make millions*. Google. https://web.dev/milliseconds-make-millions/

[2] Akamai Technologies. (2017). *Online retail performance report: Milliseconds are critical*. Akamai.

[3] Belshe, M., Peon, R., & Thomson, M. (2015). *Hypertext Transfer Protocol Version 2 (HTTP/2)* (RFC 7540). Internet Engineering Task Force.

[4] Bocchi, E., De Cicco, L., & Rossi, D. (2016). Measuring HTTP/2 performance. In *Proceedings of IEEE INFOCOM 2016* (pp. 1–9). IEEE.

[5] Facebook. (2022). *React 18 documentation*. Meta Platforms, Inc.

[6] Google. (2020). *The business impact of Core Web Vitals*. Google Developers.

a. https://web.dev/vitals-business-impact/

[7] Google. (2021). *First Input Delay (FID)*. Google Developers. https://web.dev/fid/

[8] Google. (2023a). *Core Web Vitals overview*. Google Developers. https://web.dev/vitals/

[9] Google. (2023b). *Interaction to Next Paint (INP) replacing FID*. Google Developers. https://web.dev/blog/inp-cwv-march-12

[10] Google. (2023c). *Interaction to Next Paint (INP)*. Google Developers. https://web.dev/articles/inp

[11] Google. (2023d). *Rendering performance*. Google Developers. https://web.dev/rendering-performance/

[12] Google. (2024). *Long Animation Frames API*. Google Developers. https://web.dev/long-animation-frames/

[13] Grigorik, I. (2013). *High performance browser networking*. O'Reilly Media.

[14] Osmani, A., & Stefanovsky, K. (2019). Rendering on the web. *web.dev*. https://web.dev/articles/rendering-on-the-web

[15] Newman, S. (2015). *Building microservices: Designing fine-grained systems*. O'Reilly Media.

[16] Nielsen, J. (1993). *Usability engineering*. Academic Press.

[17] Seow, S. C. (2008). *Designing and engineering time: The psychology of time perception in software*. Addison-Wesley.

[18] Souders, S. (2007). *High performance web sites: Essential knowledge for frontend engineers*. O'Reilly Media.

[19] Tilkov, S., & Vinoski, S. (2010). Node.js: Using JavaScript to build high-performance network programs. *IEEE Internet Computing, 14*(6), 80–83.

[20] Wang, X., Balasubramanian, A., Krishnamurthy, A., & Wetherall, D. (2014). How speedy is SPDY? In *Proceedings of ACM SIGCOMM 2014* (pp. 387–398). ACM.